TECHNISCHE UNIVERSITÄT DARMSTADT

Provably Secure Advanced Cryptographic Wallets

Vom Fachbereich Informatik der TU Darmstadt genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Andreas Erwig, M.Sc.
geboren in Essen, Deutschland

Darmstadt 2023

Gutachter: Prof. Sebastian Faust, Ph.D.
Prof. Sarah Meiklejohn, Ph.D.

Datum der Einreichung: 09.06.2023

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

[illegible]

(A. Erwig)

Wissenschaftlicher Werdegang

Oktober 2012 - September 2015: Bachelor of Science in Angewandter Informatik an der Ruhr-Universität Bochum.

Oktober 2015 - September 2017: Master of Science in IT-Sicherheit an der Technischen Universität Darmstadt.

September 2018 - Juli 2023: Doktorand am Lehrstuhl für Angewandte Kryptographie an der Technischen Universität Darmstadt bei Prof. Sebastian Faust.

Acknowledgments

My time as a PhD student was marked by many ups and downs, and while the former were definitely more enjoyable, the latter allowed me to grow as a researcher and to develop as a person. I would like to take a moment to thank all the people who supported me during my PhD studies and without whom the ups would not have been as enjoyable and the downs not as tolerable.

First and foremost, I would like to thank my supervisor, Sebastian Faust, who continuously supported me throughout my PhD and who gave me the freedom to choose the topics I wanted to work on as well as to travel to various conferences and schools. I am grateful for the countless nice discussions we had and for the academic and non-academic advice I received from him. I would also like to thank Sarah Meiklejohn, Iryna Gurevych, Marc Fischlin, and Felix Wolf for agreeing to serve on my disputation committee.

I appreciate all the support I received from CROSSING, which not only funded my PhD, but also offered many opportunities to collaborate with excellent researchers, to attend workshops, and to visit conferences and schools. In particular, I would like to thank Stefanie Kettler, Jacqueline Brendel, and Johannes Braun for putting so much effort into the organization of CROSSING.

I am very grateful to my co-authors for all the intense and fruitful discussions that eventually led to our joint papers. In particular, I would like to express my gratitude to Julia Hesse for patiently guiding me through my first project at a time when I painfully discovered that good ideas in research are rare. I am also very grateful to Jacqueline Wacker and Dorothee Nikolaus, who both helped me many times to handle all the administrative work that comes up during a PhD.

Naturally, I would like to thank all my colleagues for so many nice moments in the office during joint lunches, celebrating a birthday, or just having a coffee. I especially would like to mention Max, my "therapy partner" and "LiKo", with whom I spent countless hours talking about our PhD struggles during many dinners that may or may not have involved one or two bottles of beer. Without Max, my PhD time would not have been nearly as enjoyable as it was and for that I'd like to thank him. "Yo-ho-ho und 'ne Pulle voll Rum!".

Last but not least, I owe my deepest gratitude to my parents and siblings for their unconditional support throughout my entire PhD studies, and especially to

Liz, who had to endure by far the most of my frustration, but kept on supporting and encouraging me. I honestly do not know how she always managed to stay so calm and patient, while helping me through difficult times.

List of Own Publications

Peer-reviewed Publications

[5] N. A. Alkadri, P. Das, A. Erwig, S. Faust, J. Krämer, S. Riahi, and P. Struck. “Deterministic Wallets in a Quantum World”. In: *CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.* 2020, pp. 1017–1031. **Part of this thesis.**

[15] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi. “Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures”. In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II.* 2021, pp. 635–664.

[16] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno-Sanchez. “Bitcoin-Compatible Virtual Channels”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* 2021, pp. 901–918.

[67] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “The Exact Security of BIP32 Wallets”. In: *CCS ’21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021.* 2021, pp. 1020–1042. **Part of this thesis.**

[77] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. “Two-Party Adaptor Signatures from Identification Schemes”. In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I.* 2021, pp. 451–480. **Part of this thesis.**

[79] A. Erwig, S. Faust, S. Riahi, and T. Stöckert. “CommiTEE: An Efficient and Secure Commit-Chain Protocol using TEEs”. In: *IACR Cryptol. ePrint Arch.* (2020), 1486. To appear at IEEE EuroS&P 2023.

[80] A. Erwig, M. Fischlin, M. Hald, D. Helm, R. Kiel, F. Kübler, M. Kümmerlin, J. Laenge, and F. Rohrbach. “Redactable Graph Hashing, Revisited - (Extended Abstract)”. In: *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II.* 2017, pp. 398–405.

[81] A. Erwig, J. Hesse, M. Orlt, and S. Riahi. “Fuzzy Asymmetric Password-Authenticated Key Exchange”. In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II.* 2020, pp. 761–784.

[83] A. Erwig and S. Riahi. “Deterministic Wallets for Adaptor Signatures”. In: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II.* 2022, pp. 487–506. **Part of this thesis.**

Non-Refereed Articles / Articles in Submission

[66] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “BIP32-Compatible Threshold Wallets”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 312. **Part of this thesis.**

[78] A. Erwig, S. Faust, and S. Riahi. “Large-Scale Non-Interactive Threshold Cryptosystems in the YOSO Model”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1290.

My Contribution

This thesis is based on five papers that were the outcome of collaborations between myself and several excellent researchers: Nabil Alkeilani Alkadri (CISPA Helmholtz Center for Information Security), Poulami Das (CISPA Helmholtz Center for Information Security), Sebastian Faust (TU Darmstadt), Kristina Hostáková (ETH Zurich), Juliane Krämer (University Regensburg), Julian Loss (CISPA Helmholtz Center for Information Security), Monosij Maitra (Ruhr-University Bochum), Siavash Riahi (TU Darmstadt), and Patrick Struck (University Regensburg). I am grateful to all my co-authors for the collaboration. In the following, I will provide an overview over my own contributions in each of the five works.

Chapter 3 is based on two joint works with Poulami Das, Sebastian Faust, Julian Loss, and Siavash Riahi [66, 67]. In [67], Poulami, Siavash and I jointly provided a formal model of hierarchical deterministic wallets. I then mainly worked on the unforgeability proof of our generic hierarchical wallet construction as well as on the impossibility result which shows that our wallet unforgeability proof is optimal. Together with Siavash, I additionally worked on the wallet unlinkability proof. In [66], Poulami and I formally modeled the notion of threshold signatures with rerandomizable keys, and I additionally developed a threshold ECDSA construction with rerandomizable keys. I also came up with the final scheme that re-uses the key pair of our rerandomizable threshold ECDSA construction for a threshold verifiable random function, after initial discussions with Siavash, Poulami, and Sebastian. After discussing the security proof of this scheme with Siavash, I provided the full formal proof. Section 3.1.2 in Chapter 3 was largely taken verbatim from [66] with some adjustments and Section 3.2 in Chapter 3 contains parts that are taken verbatim from [66].

Chapter 4 is based on publication [5] which is a joint work with Nabil Alkeilani Alkadri, Poulami Das, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Together with Poulami, I worked on the notion of signature schemes with rerandomizable public keys and described the model of deterministic wallets as well as our generic wallet construction. Together with Poulami, Siavash, Nabil, and Patrick, I was involved in several discussions on the security proofs of our generic wallet construction and our generic lattice-based signature scheme with rerandomizable public keys. Finally, Siavash and I provided a discussion on the deployment of our concrete post-quantum secure deterministic wallet scheme instantiated with the qTesla signature scheme over blockchain networks.

Chapter 5 is based on publications [77, 83] where [83] is a joint work with Siavash Riahi, and [77] is a joint work with Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. In [83], I mostly focused on the formal modeling of the notion of adaptor signatures with rerandomizable keys as well as on the conceptual description of adaptor wallets. As a part of this, I developed several security games and defined the notion of witness rerandomizable hard relations. Together with Siavash, I showed that the existing related key attack for multiplicatively rerandomizable ECDSA signatures can be applied to pre-signatures of our multiplicatively rerandomizable ECDSA-based adaptor signature scheme. The discussion of the impossibility of independent statement/witness derivation was a joint contribution by Siavash and me. The introduction of Chapter 5 as well as Section 5.2.1 in Chapter 5 were largely taken verbatim from [83] with some adjustments. In [77], I focused mostly on the formal modeling of the notions of two-party adaptor signatures with aggregatable public keys and regular two-party signatures with aggregatable public keys. I additionally discussed with Siavash the security analysis of our generic two-party signature scheme with aggregatable public keys construction, and I formalized our discussion by writing it into a formal proof. Finally, I also helped Siavash and Kristina with the security proof for our generic construction of single party adaptor signatures.

Abstract

The introduction of Bitcoin in 2008 has sparked wide attention as the concept of a decentralized cryptographic currency seemingly promised to revolutionize the financial sector. Indeed, 15 years after Bitcoin has been introduced, there exist a myriad of decentralized cryptocurrencies with millions of users around the world. Virtually all cryptocurrencies rely on digital signatures as an authentication mechanism for payments, i.e., whenever a user issues a payment, it must attach a digital signature under its signing key so as to authorize the transaction. That is, the funds of a user in a cryptocurrency network are directly tied to the user's signing key which conversely means that the loss of the signing key directly translates to the loss of the user's funds. Cryptographic wallets have become an essential tool in the cryptocurrency space to allow users to securely store and maintain their signing keys. However, despite significant efforts to develop secure cryptographic wallets, various attacks in the past have proven that this is a tedious task, and that an insecure wallet scheme can lead to the theft of millions of USD from users.

In this thesis, we significantly contribute to the development and analysis of *provably secure* cryptographic wallets. As a first step, we provide a rigorous security analysis of the *Bitcoin Improvement Proposal 32* (BIP32), the current state of the art standard for cryptographic wallets that is widely used in practice today. We find that a simple change to the standard can significantly increase its concrete security. As a second step, we develop novel wallet schemes that improve upon the state of the art by either providing better security or functionality. More concretely, we present a threshold version of BIP32 where the signing secret key of a wallet is split among several devices. This notably increases the standard's security as it prevents a single point of failure. We then present the first ever deterministic wallet scheme that remains secure even against a quantum adversary. Finally, we present the first deterministic wallet that supports so-called adaptor signatures, an advanced signature primitive with various applications in the cryptocurrency space. We additionally extend the adaptor signature primitive to a two-party scheme, and we discuss applications of such a scheme in cryptocurrency networks. Importantly, we provide formal models as well as rigorous security proofs for all of our constructions according to the paradigm of *modern cryptography*, and we generally advocate for the use of *provably secure* cryptographic wallets in practice.

Zusammenfassung

Die Einführung von Bitcoin im Jahr 2008 erregte große Aufmerksamkeit, da das Konzept einer dezentralen Kryptowährung eine Revolution des Finanzsektors zu versprechen schien. In der Tat gibt es 15 Jahre nach der Einführung von Bitcoin eine Vielzahl dezentraler Kryptowährungen mit Millionen von Nutzern weltweit. Nahezu alle Kryptowährungen beruhen auf digitalen Signaturen, die als Authentifizierungsmechanismus für Zahlungen verwendet werden, d. h., wenn ein Nutzer eine Zahlung leistet, muss er eine digitale Signatur unter seinem Signaturschlüssel erstellen, um die Transaktion zu autorisieren. Dies bedeutet, dass die Münzen eines Benutzers in einem Kryptowährungsnetzwerk direkt an den Signaturschlüssel dieses Benutzers gebunden sind, was im Umkehrschluss bedeutet, dass der Verlust des Signaturschlüssels mit dem Verlust der Münzen des Benutzers gleichzusetzen ist. Kryptographische Wallets sind im Bereich der Kryptowährungen zu einem unverzichtbaren Instrument geworden, das es den Nutzern ermöglicht, ihre Signaturschlüssel sicher zu speichern und zu verwalten. Trotz erheblicher Bemühungen, sichere kryptographische Wallets zu entwickeln, haben allerdings verschiedene Angriffe in der Vergangenheit bewiesen, dass dies eine schwierige Aufgabe ist und dass ein unsicheres Wallet-System dazu führen kann, dass Benutzern Millionen von US-Dollar gestohlen werden.

In dieser Dissertation leisten wir einen wesentlichen Beitrag zu der Entwicklung und Analyse von beweisbar sicheren kryptographischen Wallets. In einem ersten Schritt analysieren wir ausführlich die Sicherheit des *Bitcoin Improvement Proposal 32* (BIP32), dem aktuellen Stand der Technik für kryptographische Wallets, der heutzutage in der Praxis weit verbreitet ist. Wir stellen fest, dass eine einfache Änderung des Standards dessen konkrete Sicherheit signifikant erhöhen kann. In einem zweiten Schritt entwickeln wir neue Wallet Schemata, die den Stand der Technik verbessern, indem sie entweder mehr Sicherheit oder mehr Funktionalität bieten. Konkret zeigen wir zunächst eine threshold Version von BIP32, bei der der geheime Signaturschlüssel eines Wallets auf mehrere Geräte verteilt wird. Dies erhöht die Sicherheit des Standards erheblich, da es einen Single Point of Failure vermeidet. Anschließend stellen wir das erste deterministische Wallet Schema vor, das auch gegen einen Angreifer mit Zugriff auf einen Quantencomputer sicher bleibt. Abschließend entwickeln wir das erste deterministische Wallet Schema, das die Ausführung sogenannter Adaptor-Signaturen unterstützt, ein komplexes Signaturverfahren mit verschiedenen Anwendungen im Bereich der Kryptowährungen. Darüber hinaus erweitern wir Adaptor-Signaturen zu einem Zweiparteiensystem und diskutieren Anwendungen eines solchen Systems in Kryptowährungsnetzwerken. Wir betonen, dass wir formale Modelle und Sicherheitsbeweise für alle unsere Konstruktionen nach den Richtlinien der *modernen Kryptographie* entwickeln und wir generell die Verwendung von *beweisbar sicheren* kryptographischen Wallets in der Praxis unterstützen wollen.

Contents

1. Introduction

In 2008, Satoshi Nakamoto introduced Bitcoin [146], the first ever decentralized cryptocurrency, and with it a new digital payment paradigm. A decentralized cryptocurrency differs crucially from centralized digital cash systems (e.g., [40, 51, 52, 54]) or traditional payment solutions offered by banks and other financial institutions in the sense that it does not rely on a trusted authority. A fundamental issue of digital currencies and of decentralized currencies in particular, is the problem of double spending, where users spend the same digital coin more than once, effectively copying their money. Intuitively, this issue seems difficult to overcome: since coins of digital currencies are merely data stored on an electronic device, a user can always copy the data thereby "multiplying" its money arbitrarily. In order to prevent this issue in Bitcoin, Nakamoto introduced the concept of a blockchain, a public ledger which keeps track of every single transaction that is processed in the system. Every user in the cryptocurrency network has access to the blockchain and can therefore verify if a coin has been spent previously. At a more technical level, a blockchain is a data structure consisting of "blocks" of transactions where each block is connected to the previous block via means of a cryptographic hash function. The cryptographic connection between blocks ensures the integrity of the data stored on the blockchain. Finally, in order to maintain such a data structure in a decentralized network with mutually distrusting users, Nakamoto proposes to use the so-called proof-of-work (PoW) mechanism [74]. Essentially, PoW serves as a countermeasure against Sybil attacks [72] and therefore allows all users to reach consensus on the state of the blockchain as long as the majority of the network's computing power is used honestly.

Following the introduction of Bitcoin, the concept of cryptocurrencies and the blockchain technology in general has gained wide popularity with promises to revolutionize the financial sector, as it allows users to be in control of their own money without having to adhere to restrictions and regulations of traditional banks. Indeed, 15 years after the introduction of Bitcoin, there exist a myriad of different cryptocurrencies with a combined market capitalization of more than one trillion USD[1] at the time of writing. Unfortunately, despite their wide adoption in

[1] `https://coinmarketcap.com/charts/`

practice, the technology behind cryptocurrencies is still in its infancy and has not been properly analyzed in many cases. Deploying technology that lacks a sound and extensive security analysis is dangerous, especially in the financial sector where people's livelihood may depend on the well-functioning and security of the technology. Indeed, over the years various security vulnerabilities in cryptocurrency networks have been discovered (e.g., [13, 157]) and often exploited. The most fundamental technological building block of virtually all cryptocurrency and blockchain networks is a digital signature scheme, which allows users to authenticate their transactions. Essentially, in a cryptocurrency network each user holds a signing public/secret key pair which is used to issue and receive transactions. For instance, consider a cryptocurrency user Alice holding a public/secret key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$. Alice uses her public key pk_A as an address to receive coins from other users, and she can send a payment of c coins to another user Bob, who holds public key pk_B, by (1) assembling a transaction of the form "*Transfer c coins from* pk_A *to* pk_B", and (2) generating a digital signature on the transaction using sk_A. The signature serves as an authentication mechanism that allows the cryptocurrency network to verify that it was indeed Alice who issued the transaction. The security of Alice's funds crucially relies on this authentication mechanism: assume an adversary Charlie with key pair $(\mathsf{pk}_C, \mathsf{sk}_C)$ is able to forge a valid signature under Alice's public key pk_A for the transaction "*Transfer c coins from* pk_A *to* pk_C". Then Charlie can essentially spend Alice's funds without her consent. Therefore, the digital signature scheme must guarantee the security property of *unforgeability*, which intuitively says that no adversary can generate a valid signature under a public key, for which it does not know the corresponding secret key. Given this property, the only way for an attacker to steal Alice's cryptocurrency funds is to obtain her signing secret key, meaning conversely that Alice's funds stay secure, as long as her secret key is protected from attackers. Unfortunately, in practice it is a challenging task to protect against hackers at all time, and indeed, in the year of 2022 alone, hackers managed to steal almost four billion USD [71] from cryptocurrency users of which a significant amount resulted from private key compromise and wallet vulnerabilities [50].

1.1. Cryptographic Wallets

In the blockchain and cryptocurrency setting, the storage and maintenance of users' signing keys is handled by so-called *cryptographic wallets*. Over the years, many cryptographic wallet schemes with different characteristics and designs have been introduced (e.g., [11, 68, 111, 123, 139]). Two particularly distinguishing design choices for cryptographic wallets are (1) their implementation type, and (2) their custody type. The former distinguishes whether a wallet is implemented in software (so-called software wallet) or in hardware (so-called hardware wallet). The difference between these two is essentially a trade-off between usability and security: a software wallet (e.g., Electrum [75]) might for instance run on a user's smartphone and is therefore convenient to use, but at the same time it is exposed to a high risk of corruption. A hardware wallet (e.g., Ledger [127] or Trezor [175]), on the other hand, is a special purpose hardware device, which is specifically made for the purpose of protecting the keys of a user and is therefore more resistant to attackers. On the downside, however, a user of a hardware wallet must always use a separate device and activate it in order to issue a transaction.

The second design choice is the custody type, which determines who is in control of the wallet. Broadly there are three custody types for cryptographic wallets, namely *non-custodial*, *custodial*, and *shared-custodial*. While it seems most natural that the user is in control of its own wallet, this so-called non-custodial approach has significant downsides. First, the user might lose its wallet and consequently lose all of its money. Second, the user might not have the technical expertise or infrastructure to effectively protect its wallet from attackers. Finally, users of non-custodial wallets typically rely on software or hardware developed by a third party and must therefore often trust that the respective wallet does not contain any vulnerabilities. Indeed, several attacks on different non-custodial wallets (e.g., [59, 172]) show that this is a serious risk in practice. A popular alternative to non-custodial wallets are custodial wallets, where a service provider such as Coinbase[2], Binance[3], or Fireblocks[4] operates the wallet on behalf of the user. While this is a convenient solution for the user, it requires the user to fully trust the service provider to store the keys securely and to prevent any misuse of the user's funds. Finally, shared-custodial wallets represent a middle ground between non-custodial and custodial wallets, where the user and a service provider jointly maintain the wallet such that none can use the user's secret key without the consent of the other.

Deterministic Wallets. Regardless of its implementation and custody type, it is generally recommended that a wallet scheme follows the concept of *deterministic wallets* [140, 179]. Indeed, most wallets that are widely used in practice today (e.g., Electrum [75], Ledger [127], Trezor [175]) follow this concept. At a high level, a deterministic wallet is initialized with a so-called *master* key pair $(\mathsf{pk}, \mathsf{sk})$

[2]`https://www.coinbase.com/`
[3]`https://www.binance.com/`
[4]`https://www.fireblocks.com/`

as well as a state St, and it defines two deterministic key derivation algorithms, namely one public key and one secret key derivation algorithm. These algorithms respectively allow to deterministically derive so-called *session* keys from the master key pair and the state. In more detail, the public key derivation algorithm takes as input the master public key pk, the state St, and an identifier ID and it outputs a session public key $\mathsf{pk}_{\mathsf{ID}}$. The secret key derivation algorithm works analogously for secret keys. If both algorithms are executed on the same identifier ID, then the resulting session key pair $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ constitutes a valid signing key pair. This concept of deterministic wallets has first been formalized in a work by Das et al. [68], which defines two security properties for such a wallet scheme, namely *wallet unforgeability* and *wallet unlinkability.* The former guarantees that an adversary, which knows only the master public key and the state, cannot forge a valid signature under the master public key or *any* valid session public key. The latter guarantees that an adversary cannot distinguish session public keys from independently generated public keys without knowledge of the state St.

The wallet unlinkability property is the main reason for the popularity of deterministic wallets, as it provides privacy to users in the cryptocurrency network. Recall that a transaction in a cryptocurrency specifies the sender and recipient of a payment by their respective public keys. Therefore, if a user uses the same public key for all transactions, it is easy to track and link all of this user's transactions. A trivial remedy to this problem would be to generate a fresh key pair each time the user receives a payment, which requires however to store and maintain many key pairs at the same time. Deterministic wallets offer a more elegant solution to this problem, as they must only store the master key pair and the state, and can derive session key pairs "on the fly".

The Hot/Cold Setting. Das et al. [68] formalize deterministic wallets in the so-called hot/cold setting, where the wallet scheme consists of two devices, a hot and a cold wallet device. The hot wallet device stores only the master public key pk and the state St and is permanently connected to the Internet, whereas the cold wallet stores the master secret key sk and the state St and remains offline for the majority of the time unless it must generate a signature to authenticate a transaction. The reasoning is then that it is difficult for an adversary to corrupt the cold wallet (and consequently the master secret key), as it remains offline most of the time.

Deterministic wallets are well suited to be implemented in the hot/cold setting due to their deterministic key derivation algorithms, which allow the hot and cold wallet to *independently* derive session public and secret keys. To illustrate this, let us assume a user Alice, who operates a deterministic wallet in the hot/cold setting:

in order to receive a payment, Alice instructs the hot wallet to derive a fresh session public key, say $\mathsf{pk}_{\mathsf{ID}}$, from the master public key pk and the state St. Note that this derivation does not require any interaction with the cold wallet, i.e., the cold wallet can remain offline. Only when Alice wishes to spend the coins that she received for $\mathsf{pk}_{\mathsf{ID}}$, she instructs the cold wallet to derive the corresponding secret key $\mathsf{sk}_{\mathsf{ID}}$ from sk and St and to generate a signature on the spending transaction.

1.2. Goal of this Thesis

The goal of this thesis is to advance the field of secure cryptographic wallets by (1) providing a formal security analysis of the current state of the art standard for cryptographic wallets, namely the BIP32 standard that specifies so-called hierarchical deterministic wallets [179], and (2) introducing novel wallet constructions that improve upon the state of the art by either providing better security guarantees or enhanced functionalities. In order to establish confidence in the security of our constructions, we provide rigorous security analyses of our schemes using techniques from the field of *modern cryptography*. That is, in order to prove the security of a scheme, we first formally define reasonable security properties that the scheme must satisfy as well as an adversary model that describes the influence an attacker might have on the scheme in practice. For instance, a semi-honest adversary can corrupt honest parties but not control them, i.e., it can merely observe the communication and computation that happens on the corrupted device. On the other hand, a significantly stronger adversary model considers a so-called fully malicious adversary that cannot only corrupt honest parties but also fully control them and deviate arbitrarily from the prescribed protocol instructions. Once the security model of a scheme is established, we formally prove that the scheme satisfies the defined security properties w.r.t. the adversary model. In modern cryptography, such proofs typically rely on the assumption that a certain mathematical problem is computationally hard to solve (e.g., the factorization of large numbers) or that the security of a certain cryptographic scheme is computationally hard to break. Computational hardness means that the respective assumption can *theoretically* be broken, but it is practically infeasible to do so. In this thesis, we generally provide security proofs by reduction, a well established proof technique, where we show that if there exists an adversary within our adversary model that can break the security of our scheme, then we can solve an underlying mathematical problem or break the security of a cryptographic scheme, which contradicts our assumptions. We can therefore conclude that our scheme must be secure w.r.t. our security and adversary model. It is important to emphasize that security is only guaranteed *within our security and adversary model.* Any attack or adversarial capability that might occur in practice but that is not covered by our model may render the respective scheme insecure. It is therefore crucial to define a model that closely reflects the real world.

In summary, with this thesis we aim to advance the field of *provably secure* cryptographic wallet schemes. A formal security proof carried out in an accurate security and adversary model significantly increases the confidence in the actual security of the respective scheme and can help to prevent attacks with disastrous consequences. We would therefore like to advocate for the use of provably secure cryptographic wallet schemes in practice.

1.3. Thesis Outline

In Chapter 2, we provide necessary notation that we use throughout this thesis. We additionally recall the cryptographic primitives of digital signature schemes and signature schemes with rerandomizable keys as well as the ECDSA signature scheme, since the contribution of several works included in this thesis relies on these primitives.

In Chapter 3, we detail the contribution of our two works [66, 67] on the BIP32 standard for hierarchical deterministic wallets [179]. More concretely, we first describe our contribution contained within publication [67], where we formally model hierarchical deterministic wallets according to the BIP32 standard and analyze its concrete bit-security level. We then describe the contribution of our second work [66] on the BIP32 standard, where we show how to translate the standard to the threshold setting.

Chapter 4 is based on publication [5], where we initiate the study on deterministic wallets which are secure against adversaries with access to a quantum computer. In more detail, in publication [5] we describe the model of deterministic wallets in the post-quantum setting, where we assume that the adversary has access to a quantum computer, while honest parties run on classical computers. We then show a generic construction of such a deterministic wallet and prove that the construction is secure in our model.

Chapter 5 describes our contribution contained within the two publications [77, 83]. That is, we first describe our contribution from publication [83], where we introduce the notion of adaptor wallets which are essentially deterministic wallets in the hot/cold setting that support not only standard digital signature schemes but even the more advanced notion of adaptor signatures [15, 96, 152]. We then describe our contribution from publication [77], where we first show how to generically construct adaptor signatures, and then introduce the notion of two-party adaptor signatures with aggregatable public keys, for which we also provide a generic construction.

Finally, in Chapter 6, we conclude this thesis by providing a discussion on interesting future research directions.

2. Preliminaries

In this section, we describe the notation that we use throughout the thesis, and we recall necessary cryptographic building blocks.

2.1. Notation

We denote the set of all natural numbers by $\mathbb{N}$, the set of all real numbers by $\mathbb{R}$, the set of all integers by $\mathbb{Z}$, and the set of integers $\{1, \cdots, q\}$ by $[q]$. We write $a \leftarrow_{\$} \mathsf{H}$ to denote the uniform random sampling of an element a from set H. We denote the execution of a probabilistic algorithm A that outputs y on input x by $y \leftarrow_{\$} A(x)$. Similarly, for a deterministic algorithm B, we write $y \leftarrow B(x)$ to denote that B outputs y on input x. We use the notation $y \in B(x)$ to denote that the value y lies in the set of possible outputs of algorithm B on input x. For our definitions, we implicitly assume that all algorithms receive required public parameters par as input, and we generally denote the security parameter by κ. We say that a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is *negligible*, if for all polynomials $p(\cdot)$ there exists an integer $N > 0$ such that for all $\kappa > N$ it holds that $\mathsf{negl}(\kappa) < \frac{1}{p(\kappa)}$. Throughout this thesis, we abbreviate the term *probabilistic polynomial-time* by PPT. Finally, in order to define a security property of a cryptographic scheme, we use standard code-based security games [165], which are interactive probability experiments between a challenger and an adversary.

2.2. Digital Signatures

Virtually all cryptographic wallets rely on digital signature schemes, a fundamental cryptographic building block which we recall in the following.

2.2.1. Definition of Digital Signatures

Definition 2.2.1 (Signature scheme)**.** *A signature scheme* Sig *is defined w.r.t. a message space* $\mathcal{M}$ *and consists of a triple of algorithms* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *which are defined as follows:*

- $\mathsf{Gen}(1^\kappa)$: *The probabilistic key generation algorithm* Gen *takes as input a security parameter* κ *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: *The probabilistic signing algorithm* Sign *takes as input a secret key* sk *and message* $m \in \mathcal{M}$ *and outputs a signature* σ.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *The deterministic verification algorithm* Verify *takes as input a public key* pk, *a message* $m \in \mathcal{M}$, *and a signature* σ *and outputs a bit* $b \in \{0, 1\}$. *If the output is* 1, σ *is called a valid signature.*

A signature scheme Sig *is correct if for all* $\kappa \in \mathbb{N}$, *all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(1^\kappa)$, *and all* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1\right] = 1.$$

A signature scheme must satisfy the security notion of *unforgeability under chosen message attacks*, which essentially guarantees that an adversary, which receives signatures for messages of its choice, cannot forge a valid signature for a new message without knowledge of the scheme's secret key.

Definition 2.2.2 (Unforgeability under chosen message attacks of signature schemes)**.** *A signature scheme* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\mathbf{uf{-}cma}$*-secure if for any PPT adversary* $\mathcal{A}$ *the following holds:*

$$\Pr[\mathbf{uf{-}cma}^{\mathcal{A}}_{\mathsf{Sig}}(1^\kappa) = 1] \leq \mathsf{negl}(\kappa),$$

where negl *is a negligible function in the security parameter* κ *and the game* $\mathbf{uf{-}cma}_{\mathsf{Sig}}$ *is defined in Figure 2.1.*

```
Game $\mathbf{uf{-}cma}_{\mathsf{Sig}}(1^\kappa)$
00 $\mathsf{SigList} := \emptyset$
01 $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Sig.Gen}(1^\kappa)$
02 $(\sigma^*, m^*) \leftarrow_\$ \mathcal{A}^{\mathtt{Sign}}(\mathsf{pk})$
03 $b_1 \leftarrow \mathsf{Sig.Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
04 $b_2 \leftarrow m^* \notin \mathsf{SigList}$
05 Return $b_1 \wedge b_2$

Oracle $\mathtt{Sign}(m)$
06 $\sigma \leftarrow_\$ \mathsf{Sig.Sign}(\mathsf{sk}, m)$
07 $\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{m\}$
08 Return $\sigma$
```

Figure 2.1.: Unforgeability game $\mathbf{uf{-}cma}_{\mathsf{Sig}}$ for a signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

In this thesis, we also consider the slightly weaker security notion of *one-per message unforgeability under chosen message attacks* [23, 92, 103, 153], which essentially restricts the adversary to obtain at most one signature per message before it has to output its forgery for a new message.

Definition 2.2.3 (One-per message unforgeability under chosen message attacks of signature schemes)**.** *A signature scheme* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $\mathbf{uf{-}cma1}$*-secure if for any PPT adversary* $\mathcal{A}$ *the following holds:*

$$\Pr[\mathbf{uf{-}cma1}^{\mathcal{A}}_{\mathsf{Sig}}(1^\kappa) = 1] \leq \mathsf{negl}(\kappa),$$

where negl *is a negligible function in the security parameter* κ *and the game* $\mathbf{uf{-}cma1}_{\mathsf{Sig}}$ *is defined in Figure 2.2.*

```
Game $\mathbf{uf{-}cma1}_{\mathsf{Sig}}(1^\kappa)$
00 $\mathsf{SigList} := \emptyset$
01 $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Sig.Gen}(1^\kappa)$
02 $(\sigma^*, m^*) \leftarrow_\$ \mathcal{A}^{\mathtt{Sign}}(\mathsf{pk})$
03 $b_1 \leftarrow \mathsf{Sig.Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
04 $b_2 \leftarrow m^* \notin \mathsf{SigList}$
05 Return $b_1 \wedge b_2$

Oracle $\mathtt{Sign}(m)$
06 If $m \in \mathsf{SigList}$: Return $\perp$
07 $\sigma \leftarrow_\$ \mathsf{Sig.Sign}(\mathsf{sk}, m)$
08 $\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{m\}$
09 Return $\sigma$
```

Figure 2.2.: One-per message unforgeability game $\mathbf{uf{-}cma1}_{\mathsf{Sig}}$ for a signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

2.2.2. The ECDSA Signature Scheme

As several works in this thesis rely on the elliptic curve digital signature algorithm (ECDSA) [116], we briefly recall the scheme here. The ECDSA signature scheme is the elliptic curve variant of the digital signature algorithm [124] which was first proposed by Kravitz in 1991 and later standardized by the NIST. The ECDSA signature scheme is defined w.r.t. a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q and with base point G, where $\mathbb{G}$ is defined as the group of points on an elliptic curve. It must hold that the discrete logarithm problem is hard in $\mathbb{G}$. We additionally denote the group $\mathbb{G}$ that does not include the point at infinity O by $\mathbb{G}^* := \mathbb{G} \setminus \{O\}$. In this thesis, we use additive notation for the group operation, and we note that any point $P \in \mathbb{G}$ can be expressed as $P = x \cdot G$ for $x \in \mathbb{Z}_q$. The ECDSA scheme internally uses a cryptographic hash function $\mathsf{H}_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and a function

$f : \mathbb{G}^* \to \mathbb{Z}_q$ which is the projection of a group element to its x-coordinate. We denote the ECDSA scheme by $\mathsf{EC}[\mathsf{H}_0]$ and recall the full scheme in Figure 2.3.

```
$\mathsf{Gen}(1^\kappa)$
$x \leftarrow_\$ \mathbb{Z}_q$
$X \leftarrow x \cdot G$
$(\mathsf{sk}, \mathsf{pk}) := (x, X)$
Return $(\mathsf{sk}, \mathsf{pk})$
```

```
$\mathsf{Sign}(\mathsf{sk}, m)$
$k \leftarrow_\$ \mathbb{Z}_q, R \leftarrow k \cdot G$
$r \leftarrow f(R)$
$h \leftarrow \mathsf{H}_0(m)$
$s = k^{-1}(h + r \cdot x)$
$\sigma := (r, s)$
Return $\sigma$
```

```
$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
Parse $\mathsf{pk} := X$
Parse $\sigma := (r, s)$
If $s = 0 \vee t = 0$: Return $\perp$
$h \leftarrow \mathsf{H}_0(m)$
$u_1 \leftarrow h \cdot s^{-1}$
$u_2 \leftarrow r \cdot s^{-1}$
$R \leftarrow u_1 \cdot G + u_2 \cdot X$
If $f(R) = r$: Return 1
Return 0
```

Figure 2.3.: ECDSA signature scheme $\mathsf{EC}[\mathsf{H}_0]$ instantiated with a hash function $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_q$. The function $f : \mathbb{G}^* \to \mathbb{Z}_q$ is the projection of a group element to its x-coordinate.

2.3. Signature Schemes with Rerandomizable Keys

We briefly recall the notion of *signature schemes with rerandomizable keys*. This notion has first been introduced by Fleischhacker et al. [94] and has been shown to be a useful primitive for the construction of cryptographic wallet schemes (e.g., [66, 67, 68]). At a high level, a signature scheme with rerandomizable keys extends the notion of standard signature schemes by two deterministic key derivation algorithms, which respectively allow to rerandomize the scheme's secret and public key.

Definition 2.3.1 (Signature scheme with rerandomizable keys)**.** *A signature scheme with rerandomizable keys* RSig *consists of a tuple of algorithms* RSig = (Gen, Sign, Verify, RandSK, RandPK*) where* (Gen, Sign, Verify) *are the standard algorithms of a signature scheme (cf. Definition 2.2.1). Moreover, let the public parameters* par *define a randomness space* $\mathcal{R} := \mathcal{R}(\mathsf{par})$*. Then the algorithms* RandSK *and* RandPK *are defined as follows:*

- RandSK(sk, ρ)*: The deterministic* secret key rerandomization algorithm RandSK *takes as input a secret key* sk *and randomness* $\rho \in \mathcal{R}$ *and outputs a rerandomized secret key* sk′*.*

- $\mathsf{RandPK}(\mathsf{pk}, \rho)$*: The deterministic* public key rerandomization algorithm RandPK *takes as input a public key* pk *and randomness* $\rho \in \mathcal{R}$ *and outputs a rerandomized public key* pk'*.*

We make the convention that for the empty string ϵ it holds $\mathsf{RandPK}(\mathsf{pk}, \epsilon) = \mathsf{pk}$ and $\mathsf{RandSK}(\mathsf{sk}, \epsilon) = \mathsf{sk}$. We further require that a signature scheme with rerandomizable keys satisfies *(perfect) rerandomizability of keys* and *correctness under rerandomized keys*:

1. *(Perfect) rerandomizability of keys:* For all $\kappa \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\kappa)$ and $\rho \leftarrow_\$ \mathcal{R}$, the distributions of $(\mathsf{pk}', \mathsf{sk}')$ and $(\mathsf{pk}'', \mathsf{sk}'')$ are identical, where:

$$(\mathsf{pk}', \mathsf{sk}') \leftarrow (\mathsf{RandPK}(\mathsf{pk}, \rho), \mathsf{RandSK}(\mathsf{sk}, \rho)) \text{ and } (\mathsf{pk}'', \mathsf{sk}'') \leftarrow_\$ \mathsf{Gen}(1^\kappa).$$

2. *Correctness under rerandomized keys:* For all $\kappa \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\kappa)$, all $\rho \in \mathcal{R}$, and all $m \in \{0,1\}^*$, the rerandomized keys $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$ satisfy:

$$\Pr[\mathsf{Verify}(\mathsf{pk}', \sigma, m) = 1 \mid \sigma \leftarrow_\$ \mathsf{Sign}(\mathsf{sk}', m)] = 1.$$

A signature scheme with rerandomizable keys must satisfy the security notion of *unforgeability under honestly rerandomizable keys.* This notion differs from the unforgeability notion of standard signature schemes in the following ways: First, the adversary can receive signatures for messages of its choice under honestly rerandomized keys, i.e., under keys that have been rerandomized with an honestly sampled randomness. Second, the adversary can output a valid forgery for a new message under *any* honestly rerandomized key.

Definition 2.3.2 (Unforgeability under honestly rerandomizable keys)**.** *A signature scheme with rerandomizable keys* $\mathsf{RSig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* $\mathbf{uf{-}cma{-}hrk}$*-secure if for any PPT adversary* $\mathcal{A}$ *the following holds:*

$$\Pr[\mathbf{uf{-}cma{-}hrk}^{\mathcal{A}}_{\mathsf{RSig}} = 1] \leq \mathsf{negl}(\kappa),$$

where negl *is a negligible function in the security parameter* κ *and the game* $\mathbf{uf{-}cma{-}hrk}_{\mathsf{RSig}}$ *is defined in Figure 2.4.*

Similarly to standard signature schemes, we also consider the slightly weaker notion of *one-per message unforgeability under honestly rerandomizable keys* for signature schemes with rerandomizable keys.

```
Game $\mathbf{uf{-}cma{-}hrk}_{\mathsf{RSig}}(1^\kappa)$
$\mathsf{SigList} := \emptyset$, $\mathsf{RList} \leftarrow \{\epsilon\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{RSig.Gen}(1^\kappa)$
$(\sigma^*, m^*, \rho^*) \leftarrow_\$ \mathcal{A}^{\texttt{Rand},\texttt{RSign}}(\mathsf{pk})$
$b_1 \leftarrow \rho^* \in \mathsf{RList}$
$\mathsf{pk}^* \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}, \rho^*)$
$b_2 \leftarrow \mathsf{RSig.Verify}(\mathsf{pk}^*, m^*, \sigma^*) = 1$
$b_3 \leftarrow m^* \notin \mathsf{SigList}$
Return $b_1 \wedge b_2 \wedge b_3$

Oracle $\texttt{Rand}(m)$
$\rho \leftarrow_\$ \mathcal{R}$
$\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
Return $\rho$

Oracle $\texttt{RSign}(m, \rho)$
If $\rho \notin \mathsf{RList}$: Return $\perp$
$\mathsf{sk}' \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}, \rho)$
$\sigma \leftarrow_\$ \mathsf{RSig.Sign}(\mathsf{sk}', m)$
$\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{m\}$
Return $\sigma$
```

Figure 2.4.: Unforgeability under honestly rerandomizable keys game $\mathbf{uf{-}cma{-}hrk}_{\mathsf{RSig}}$ for a signature scheme with rerandomizable keys $\mathsf{RSig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RandSK}, \mathsf{RandPK})$.

```
Game $\mathbf{uf{-}cma{-}hrk1}_{\mathsf{RSig}}(1^\kappa)$
$\mathsf{SigList} := \emptyset$, $\mathsf{RList} \leftarrow \{\epsilon\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{RSig.Gen}(1^\kappa)$
$(\sigma^*, m^*, \rho^*) \leftarrow_\$ \mathcal{A}^{\texttt{Rand},\texttt{RSign}}(\mathsf{pk})$
$b_1 \leftarrow \rho^* \in \mathsf{RList}$
$\mathsf{pk}^* \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}, \rho^*)$
$b_2 \leftarrow \mathsf{RSig.Verify}(\mathsf{pk}^*, m^*, \sigma^*) = 1$
$b_3 \leftarrow (\rho^*, m^*) \notin \mathsf{SigList}$
Return $b_1 \wedge b_2 \wedge b_3$

Oracle $\texttt{Rand}(m)$
$\rho \leftarrow_\$ \mathcal{R}$
$\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
Return $\rho$

Oracle $\texttt{RSign}(m, \rho)$
If $\rho \notin \mathsf{RList}$: Return $\perp$
If $(\rho^*, m^*) \in \mathsf{SigList}$ Return $\perp$
$\mathsf{sk}' \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}, \rho)$
$\sigma \leftarrow_\$ \mathsf{RSig.Sign}(\mathsf{sk}', m)$
$\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{(\rho, m)\}$
Return $\sigma$
```

Figure 2.5.: One-per message unforgeability under honestly rerandomizable keys game $\mathbf{uf{-}cma{-}hrk1}_{\mathsf{RSig}}$ for a signature scheme with rerandomizable keys $\mathsf{RSig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RandSK}, \mathsf{RandPK})$.

Definition 2.3.3 (One-per message unforgeability under honestly rerandomizable keys)**.** *A signature scheme with rerandomizable keys* $\mathsf{RSig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* $\mathbf{uf{-}cma{-}hrk1}$*-secure if for any PPT adversary* $\mathcal{A}$ *the following holds:*

$$\Pr[\mathbf{uf{-}cma{-}hrk1}^{\mathcal{A}}_{\mathsf{RSig}} = 1] \leq \mathsf{negl}(\kappa),$$

where negl *is a negligible function in the security parameter* κ *and the game* $\mathbf{uf{-}cma{-}hrk1}_{\mathsf{RSig}}$ *is defined in Figure 2.5.*

3. The BIP32 Standard for Hierarchical Deterministic Wallets

The Bitcoin Improvement Proposal 32 (BIP32) [179] is a specification for so-called *hierarchical deterministic wallets*, which has first been introduced in 2012. Since then, the BIP32 standard has gained increasing popularity and is now widely followed by several popular wallets such as Electrum [75], Ledger [127], or Trezor [175]. In a nutshell, hierarchical deterministic wallets as proposed by the BIP32 standard extend the notion of deterministic wallets by allowing to establish a hierarchy among session keys. More concretely, a hierarchical deterministic wallet is organized in a tree structure, where each node in the tree represents a wallet instance which stores a session signing key and which is uniquely identified by an identifier ID. In hierarchical deterministic wallets, the identifier specifies the position of the particular wallet instance in the tree. Similarly to deterministic wallets, the root node is initialized with a master key pair $(\mathsf{pk}, \mathsf{sk})$ and a state St, and it can use these values to deterministically derive session signing keys for child nodes. However, in contrast to deterministic wallets, a hierarchical wallet allows child wallets to derive further children themselves, thereby creating a hierarchy of wallet instances.

BIP32 specifies the signature scheme to be used as ECDSA (cf. Section 2.2.2) since Bitcoin and many other blockchain and cryptocurrency networks support this signature scheme. Recall that the ECDSA signature scheme is defined w.r.t. an elliptic curve group $\mathbb{G}$ of prime order q and with base point G. An ECDSA public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ is then simply a discrete logarithm (dlog) instance, i.e., $\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow \mathsf{sk} \cdot G$. In order to derive child wallets, BIP32 specifies a so-called *non-hardened* child derivation mechanism which proceeds as follows: Assume a parent wallet with identifier ID, key pair $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$, and state $\mathsf{St}_{\mathsf{ID}}$ wishes to derive a child with identifier ID'. Then the parent wallet first computes $(\rho_{\mathsf{ID}'}, \mathsf{St}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}')$ and then derives the child keys as $\mathsf{pk}_{\mathsf{ID}'} \leftarrow \mathsf{pk}_{\mathsf{ID}} + \rho_{\mathsf{ID}'} \cdot G$ and $\mathsf{sk}_{\mathsf{ID}'} \leftarrow \mathsf{sk}_{\mathsf{ID}} + \rho_{\mathsf{ID}'} \mod q$, where H is a cryptographic hash function. Note that this derivation mechanism allows to derive the child public key

$\mathsf{pk}_{\mathsf{ID}'}$ using only the parent public key $\mathsf{pk}_{\mathsf{ID}}$ and state $\mathsf{St}_{\mathsf{ID}}$, i.e., it does not require the parent secret key $\mathsf{sk}_{\mathsf{ID}}$. This makes the mechanism especially useful for use in combination with the hot/cold setting where the cold wallet stores $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}})$ and the hot wallet stores $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}})$, since the parent cold wallet does not need to come online in order to derive a child public key. Unfortunately, the non-hardened child derivation suffers from a significant security issue: assume an adversary corrupts the child secret key $\mathsf{sk}_{\mathsf{ID}'}$ and additionally knows the parent public key $\mathsf{pk}_{\mathsf{ID}}$ and state $\mathsf{St}_{\mathsf{ID}}$. Then the adversary can simply compute the parent secret key $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{sk}_{\mathsf{ID}'} - \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}') \mod q$. The adversary can recursively repeat this attack until it is eventually able to compute the master secret key sk and thereby corrupt the entire wallet tree. Due to this issue, BIP32 specifies an additional child derivation mechanism, the so-called *hardened* derivation, which works exactly as the non-hardened derivation with the only difference that the value $\rho_{\mathsf{ID}'}$ is computed as $\rho_{\mathsf{ID}'} \leftarrow \mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}')$. That is, instead of using $\mathsf{pk}_{\mathsf{ID}}$ as input to H, the hardened derivation uses the parent secret key $\mathsf{sk}_{\mathsf{ID}}$. This simple change prevents the above attack as the adversary now cannot compute $\rho_{\mathsf{ID}'}$ anymore as it does not know the parent secret key $\mathsf{sk}_{\mathsf{ID}}$. The drawback of the hardened derivation, however, is that the derivation of a hardened public key always requires the parent secret key, i.e., the parent cold wallet must come online just for the derivation of the child public key.

Despite its popularity, the BIP32 standard has never been formally analyzed for the exact security guarantees that it provides prior to this thesis. A previous work by Das et al. [68] formally analyzed the notion of deterministic wallets in the hot/cold setting, and showed a generic construction from signature schemes with rerandomizable keys (cf. Definition 2.3.1). However, the work of Das et al. has three important limitations: (1) it instantiates their generic construction with a *multiplicatively* rerandomizable ECDSA signature scheme, i.e., instead of deriving a child key pair by *adding* the value ρ to the parent secret key (and $\rho \cdot G$ to the parent public key), the work of Das et al. randomizes the parent keys by *mutliplying* ρ to the keys; (2) Das et al. do not consider the hierarchical setting, i.e, child nodes cannot derive further children; and (3) Das et al. consider only non-hardened derivation of keys.

3.1. Our Contribution

In this thesis, we significantly contribute towards a better understanding of the BIP32 standard and the concrete security it provides. As a first contribution, we provide a formal model and security analysis of the BIP32 standard *as is* within the following publication that can be found in Appendix A:

[67] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “The Exact Security of BIP32 Wallets”. In: *CCS ’21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021.* 2021, pp. 1020–1042. **Part of this thesis.**

Concretely, in the above publication we address the shortcomings of the work of Das et al. [68] by (1) proving an additively rerandomizable variant of the ECDSA signature scheme one-per message unforgeable under honestly rerandomizable keys (cf. Definition 2.3.3); (2) formally modeling the BIP32 standard as is; and (3) analyzing the standard’s concrete security, and showing that our security analysis is optimal. We will discuss our contribution in more detail in Section 3.1.1.

As a second contribution, we extend the BIP32 standard to the threshold setting within the following work that can be found in Appendix B:

[66] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “BIP32-Compatible Threshold Wallets”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 312. **Part of this thesis.**

In more detail, we investigate the possibility of securing non-hardened secret keys from corruption by “thresholdizing” non-hardened nodes. That is, we assume the threshold setting where each non-hardened node consists of several devices and each device holds only a *share* of the node’s secret key. In this setting, the secret key of a non-hardened node remains secure, as long as at most a subset of its devices is corrupted, i.e., each non-hardened node can tolerate corruption up to a certain threshold. In order to construct such BIP32-compatible threshold wallets, we first introduce the notion of *threshold signature schemes with rerandomizable keys*, and we instantiate it with an additively rerandomizable threshold ECDSA scheme. We then present non-hardened and hardened derivation mechanisms for the threshold setting. We detail our contribution of this work in Section 3.1.2.

3.1.1. The Exact Security of BIP32 Wallets

In this section, we describe the contribution of our publication [67] (cf. Appendix A) in more detail. As mentioned above, our contribution in this publication is threefold, namely we (1) analyze the security of an additively rerandomizable ECDSA scheme, (2) formally model BIP32 wallets, and (3) analyze the exact security that BIP32 wallets instantiated with our additively rerandomizable ECDSA scheme achieve.

Additively Rerandomizable ECDSA

Das et al. [68] showed that a multiplicatively rerandomizable variant of ECDSA satisfies the notion of *unforgeability under honestly rerandomizable keys* (namely $\mathbf{uf{-}cma{-}hrk}$ security, cf. Definition 2.3.2). Recall that ECDSA is defined w.r.t. to an elliptic curve group $\mathbb{G}$ of prime order q and base point G. Then the multiplicatively rerandomizable ECDSA scheme of Das et al. allows to rerandomize the initial key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathbb{G} \times \mathbb{Z}_q$ with a randomness $\rho \in \mathbb{Z}_q$ by *multiplying* ρ to pk and sk respectively, i.e., it computes a rerandomized public/secret key pair $(\mathsf{pk}', \mathsf{sk}')$ as $\mathsf{pk}' \leftarrow \rho \cdot \mathsf{pk}$ and $\mathsf{sk}' \leftarrow \rho \cdot \mathsf{sk} \mod q$. We denote the multiplicatively rerandomizable ECDSA scheme of Das et al. by $\mathsf{MREC}[\mathsf{H}_0]$, and we denote the standard ECDSA scheme by $\mathsf{EC}[\mathsf{H}_1]$, where H_0 and H_1 denote the hash functions used in the respective scheme. Das et al. prove the $\mathsf{MREC}[\mathsf{H}_0]$ scheme $\mathbf{uf{-}cma{-}hrk}$ secure via reduction to the unforgeability property (namely $\mathbf{uf{-}cma}$ security, cf. Definition 2.2.2) of the $\mathsf{EC}[\mathsf{H}_1]$ scheme. The reduction crucially relies on a so-called related key attack (RKA) that allows to transform a valid signature σ under public key pk and message m to a signature σ' which is valid under a *related* public key $\mathsf{pk}' \leftarrow \rho \cdot \mathsf{pk}$ and a message m', where $\rho = \frac{\mathsf{H}_1(m')}{\mathsf{H}_0(m)}$. Intuitively, this RKA is required in the reduction to simulate the signing oracle of game $\mathbf{uf{-}cma{-}hrk}$ by transforming signatures obtained from the signing oracle of the $\mathbf{uf{-}cma}$ game to be valid under the respective rerandomized public key.

In our work, we aim to analyze the BIP32 standard as is, i.e., we must first prove the security of an additvely rerandomizable ECDSA scheme, where the initial key pair is rerandomized by *adding* randomness ρ to sk and $\rho \cdot G$ to pk. We denote this additive rerandomizable ECDSA scheme by $\mathsf{REC}[\mathsf{H}_2]$, where H_2 denotes the cryptographic hash function used by the scheme. Naturally, in order to prove the security of $\mathsf{REC}[\mathsf{H}_2]$, we must find a novel RKA that our reduction can use to transform signatures in the additive setting. Indeed, we show that a valid ECDSA signature $\sigma := (r, s)$ under public key pk and on message m, is also valid under the rerandomized public key $\mathsf{pk}' \leftarrow \rho \cdot G + \mathsf{pk}$ on message m' if ρ satisfies $\rho = \frac{\mathsf{H}_1(m) - \mathsf{H}_2(m')}{r}$. While this RKA works, it has an important drawback as compared to the RKA of Das et al.: The randomness ρ depends on the value r (which is part of signature σ) and on message m'. As a consequence, we can transform at most one signature per randomness/message pair (ρ, m'), and this is why we can prove the scheme $\mathsf{REC}[\mathsf{H}_2]$ only one-per message unforgeable under honestly rerandomizable keys (namely $\mathbf{uf{-}cma{-}hrk1}$ secure, cf. Definition 2.3.3). While $\mathbf{uf{-}cma{-}hrk1}$ security is a weaker notion than $\mathbf{uf{-}cma{-}hrk}$ security, it is sufficient for the setting of cryptographic wallets, since transactions in cryptocurrency networks have a unique identifier, i.e., a wallet never signs the same transaction twice.

With the RKA in place, we then show a reduction of the **uf−cma−hrk1** security of scheme $\mathsf{REC}[\mathsf{H}_2]$ to the **uf−cma1** security (cf. Definition 2.2.3) of scheme $\mathsf{EC}[\mathsf{H}_1]$. However, our reduction incurs a loss in the number of queries to the `Rand` oracle and therefore is *non-tight*.

Formal Model of BIP32 Wallets

As a second step, we formally model hierarchical deterministic wallets according to the BIP32 standard. In our work, we define a hierarchical deterministic wallet scheme HDWAL as a tuple of seven algorithms $\mathsf{HDWAL} = (\mathsf{Setup}, \mathsf{SKDer}_\mathsf{H}, \mathsf{SKDer}_\mathsf{NH}, \mathsf{PKDer}_\mathsf{H}, \mathsf{PKDer}_\mathsf{NH}, \mathsf{Sign}, \mathsf{Verify})$, which at a high level provide the following functionality: The Setup algorithm initializes the scheme by generating a master key pair $(\mathsf{pk}, \mathsf{sk})$ and a state St for the master wallet. The algorithms $(\mathsf{SKDer}_\mathsf{H}, \mathsf{SKDer}_\mathsf{NH}, \mathsf{PKDer}_\mathsf{H}, \mathsf{PKDer}_\mathsf{NH})$ are the respective algorithms for the derivation of hardened/non-hardened child nodes. More concretely, assume a wallet with identifier ID, key pair $(\mathsf{pk}_\mathsf{ID}, \mathsf{sk}_\mathsf{ID})$, and state St_ID wishes to derive a hardened child node with identifier ID'. Then, it can execute the algorithm $\mathsf{SKDer}_\mathsf{H}$ on input $(\mathsf{sk}_\mathsf{ID}, \mathsf{St}_\mathsf{ID}, \mathsf{ID}')$, which outputs the hardened secret key $\mathsf{sk}_{\mathsf{ID}'}$ as well as a state $\mathsf{St}_{\mathsf{ID}'}$. In order to derive the corresponding child public key, the parent wallet can execute algorithm $\mathsf{PKDer}_\mathsf{H}$ on input $(\mathsf{pk}_\mathsf{ID}, \mathsf{St}_\mathsf{ID}, \mathsf{ID}')$. The algorithms $\mathsf{SKDer}_\mathsf{NH}$ and $\mathsf{PKDer}_\mathsf{NH}$ are defined analogously for the derivation of a non-hardened child node. Finally, the algorithms Sign and Verify are the standard signing and verification algorithms of a digital signature scheme. We then say that a hierarchical deterministic wallet scheme is correct, if a hardened (non-hardened respectively) key pair derived by algorithms $\mathsf{SKDer}_\mathsf{H}$ and $\mathsf{PKDer}_\mathsf{H}$ ($\mathsf{SKDer}_\mathsf{NH}$ and $\mathsf{PKDer}_\mathsf{NH}$ respectively) w.r.t. the same identifier constitutes a valid signing key pair.

As mentioned in the introduction of this thesis, a (hierarchical) deterministic wallet scheme must satisfy two security properties, namely wallet unlinkability and wallet unforgeability. Before we describe these two properties for BIP32 wallets, we describe our adversary model and the general capabilities that the adversary has over a hierarchical deterministic wallet scheme in our model. Generally, we assume in our work that non-hardened wallet instances are implemented in the hot/cold setting and that cold wallets are incorruptible by the adversary. Therefore, we assume that the hot wallet of a non-hardened node stores the wallet's public key and state, whereas the cold wallet stores the corresponding secret key and state. Hardened wallet instances, on the other hand, are not implemented in the hot/cold setting, since the corruption of a hardened secret key has no impact on the security of the remaining wallet instances in the tree. We generally assume in our model that hardened wallets represent leafs in the wallet tree, i.e., we assume that hardened wallets cannot derive further child nodes.

Adversary Model. In our model, we consider a PPT adversary that can corrupt devices and, upon corruption, has full control over the device. With respect to a hierarchical deterministic wallet scheme, we assume that the adversary can decide the structure of the wallet tree, i.e., after initialization of the master wallet, the adversary can arbitrarily create non-hardened and hardened child wallets for adaptively chosen identifiers. The adversary can additionally request signatures for adaptively chosen messages from any wallet instance in the tree. We assume that the adversary can corrupt the hot wallet of non-hardened nodes which stores the wallet's public key and state, but not the corresponding cold wallet. Finally, the adversary can fully corrupt hardened nodes, i.e., it can even learn the hardened node's secret key. We note that we allow the adversary to adaptively decide which nodes to corrupt.

Wallet Unlinkability. Intuitively, the notion of wallet unlinkability guarantees that (hardened or non-hardened) child public keys derived from a master public key pk are computationally indistinguishable from (hardened or non-hardened) child public keys derived from an independently generated master public key pk'.

In order to formalize this notion, we describe a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ where the challenger initially executes the Setup algorithm to generate the key pair $(\mathsf{pk}, \mathsf{sk})$ and state St of the master wallet. The adversary receives pk as input, and has all the capabilities as described above in the adversary model. Eventually, the adversary outputs an identity ID^* upon which the challenger responds with a public key which is either (1) the public key $\mathsf{pk}_{\mathsf{ID}^*}$ derived from pk, St and ID^*, or (2) a public key derived from a freshly generated master public key and state as well as a random identity. The adversary wins the game if it can distinguish these two cases with more than negligible probability. Note however that, according to our adversary model, $\mathcal{A}$ is allowed to corrupt the hot wallets of non-hardened nodes and thereby to learn their public keys and states. Naturally, knowing these values, the adversary can trivially compute any non-hardened child public key that is derived from the corrupted hot wallet. Therefore, to prevent the adversary from trivially winning in case ID^* identifies a non-hardened wallet instance, the game requires that there exists no corrupted parent hot wallet in the tree hierarchy for the wallet instance with identity ID^*.

Wallet Unforgeability. The notion of wallet unforgeability guarantees that an adversary cannot forge a valid signature for (1) any uncorrupted hardened wallet instance in the tree, and (2) any non-hardened wallet instance, even if its hot wallet is corrupted.

We again formalize this notion via a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which proceeds as follows: In the beginning of the game, the challenger executes the Setup algorithm to generate the master key pair $(\mathsf{pk}, \mathsf{sk})$ and state St. The adversary receives pk and St as input, and has all the capabilities as described above in the adversary model, in particular it can request signatures for arbitrary messages from any wallet instance in the tree. The adversary wins the game if (1) it can output a valid signature for a message m^* for an uncorrupted hardened wallet, or any non-hardened wallet, and (2) the adversary has not previously requested a signature for message m^* from this specific wallet instance.

In this thesis, we consider a slightly weaker notion of wallet unforgeability for BIP32 wallets, namely *one-per message wallet unforgeability*, where the game is defined exactly as described above, with the only difference that the adversary may at most request one signature per wallet instance for each message. We emphasize again that this security notion is sufficient in the cryptocurrency setting, since transactions are unique and therefore never signed twice by a wallet. We denote the one-per message wallet unforgeability game by **wufcma1**.

The Concrete Security of BIP32 Wallets

As a final contribution, we show how to generically construct hierarchical deterministic wallets according to the BIP32 standard from any signature scheme with rerandomizable keys, and formally analyze the security of our construction. To this end, we provide formal proofs that our generic construction satisfies wallet unlinkability and one-per message wallet unforgeability. Here, we focus on the proof of one-per message wallet unforgeability, and we refer to Appendix A for the wallet unlinkability proof. As we will explain below, our wallet unforgeability proof incurs a polynomial loss in the number of corrupted hardened wallet instances. However, we show that this loss is optimal. Finally, we compute the concrete security level our construction achieves when instantiated with our additively and Das et al.'s [68] multiplicatively rerandomizable ECDSA scheme respectively.

One-Per Message Wallet Unforgeability. We show that our generic hierarchical deterministic wallet construction is one-per message wallet unforgeable if the underlying signature scheme with rerandomizable keys is one-per message unforgeable as per Definition 2.3.3. To do so, we show that if there exists an adversary $\mathcal{A}$ that wins game **wufcma1** with more than negligible probability, then we can construct an adversary $\mathcal{C}$ which uses $\mathcal{A}$ to win game **uf−cma−hrk1** with more than negligible probability. At a high level, our proof proceeds as follows: In the beginning, the adversary $\mathcal{C}$ receives a public key $\mathsf{pk}_{\mathcal{C}}$ from game **uf−cma−hrk1**, which it uses as master public key in game **wufcma1**[1]. $\mathcal{C}$ must then simulate game **wufcma1** to $\mathcal{A}$ w.r.t. $\mathsf{pk}_{\mathcal{C}}$. The main difficulty here is to simulate the corruption of a hardened wallet instance, since it requires $\mathcal{C}$ to output the hardened wallet's secret key that was correctly derived from the master secret key. However, since $\mathcal{C}$ does not know the master secret key, it cannot provide the correct hardened wallet's secret key. Therefore, we handle hardened wallet corruptions as follows: $\mathcal{C}$ guesses in advance which of the hardened nodes in the tree will get corrupted by $\mathcal{A}$ throughout the game. For these nodes, $\mathcal{C}$ generates a fresh key pair independently of $\mathsf{pk}_{\mathcal{C}}$, such that it can answer corruption queries. We show that $\mathcal{A}$ does not realize that the keys of the corrupted hardened nodes have been generated independently of $\mathsf{pk}_{\mathcal{C}}$ by providing a reduction to the one-per message unforgeability of the underlying signature scheme with rerandomizable keys.

While this guessing introduces a polynomial loss in the number of corrupted hardened nodes, it suffices to show that $\mathcal{C}$ can use $\mathcal{A}$ to win game **uf−cma−hrk1** with non-negligible probability.

On the Optimality of our Wallet Unforgeability Proof. As mentioned above, our wallet unforgeability proof incurs a polynomial loss in the number of corrupted hardened wallet instances. We show, however, that this loss is indeed inherent, i.e., we show that there exists no reduction from the **uf−cma−hrk** security of a signature scheme with rerandomizable keys RSig to the **wufcma1** security of our generic hierarchical deterministic wallet construction instantiated with RSig.[2] We do so, by using the meta-reduction technique as first introduced by Coron [61]. That is, we show that if there exists a reduction $\mathcal{R}$ that reduces the **uf−cma−hrk** security of RSig to the **wufcma1** security of our generic hierarchical deterministic wallet construction instantiated with RSig while incurring less than a polynomial loss in the number of corrupted hardened wallet instances, then we can show a meta-reduction that uses $\mathcal{R}$ to successfully break the **uf−cma−hrk** security of RSig. Our proof closely follows the proofs from Coron [61, Theorem 4] and Kakvi

[1]To be exact, $\mathcal{C}$ uses a derivation of $\mathsf{pk}_{\mathcal{C}}$ as master public key. We refer to Appendix A for more details.

[2]To be exact, we show that this result even holds for signature schemes with rerandomizable keys that satisfy the stronger **uf−cma−rk** security. For simplicity, we describe our result here w.r.t. **uf−cma−hrk** security. We refer to Appendix A for more details.

et al. [117, Theorem 2], with adaptions to our hierarchical deterministic wallet model.

Concrete Security of BIP32. Finally, we instantiate our generic hierarchical deterministic wallet construction with our additively rerandomizable ECDSA scheme and compute the exact bit security level that this instantiation provides. In order to do so, we make the following assumptions: We assume a security parameter under which the standard ECDSA scheme has a bit security of 128 bits, and we estimate the total number of keys derived in the tree hierarchy to be 2^{20} and that roughly 1% of these keys are corrupted hardened keys, i.e., we assume that 2^{14} hardened wallets get corrupted. Under these assumptions, we show that the BIP32 standard *as is* provides a bit security level of 91 bits, i.e., it loses 37 bits of security compared to the standard ECDSA signature scheme. This loss comes from (1) the loss in our security proof of the additively rerandomizable ECDSA scheme, and (2) the loss in the security proof of our generic hierarchical deterministic wallet construction. Note that, while we proved that the latter is unavoidable, we can get a better bit security level, if we instantiate our generic construction with the multiplicatively rerandomizable ECDSA scheme from Das et al. [68], since this scheme can be proven unforgeable without polynomial loss. Indeed, when instantiating our generic construction with the multiplicatively rerandomizable ECDSA scheme, our construction achieves a bit security level of 111 bits, i.e., 20 bits more than the instantiation from our additively rerandomizable ECDSA. We therefore advocate for changing the BIP32 standard to use multiplicatively rerandomizable ECDSA, as it offers significantly stronger security at virtually no efficiency loss.

3.1.2. BIP32-Compatible Threshold Wallets[3]

As explained in the introduction of this chapter, it is crucial to protect non-hardened secret keys from corruption, since the leakage of a single non-hardened secret key breaks the security of the entire BIP32 wallet scheme. In our formal analysis of BIP32 wallets above, we assume that non-hardened wallets are implemented in the hot/cold setting and that cold wallets are incorruptible. However, this assumption might not hold when considering, e.g., an adversary that obtains physical access to the cold wallet or when the cold wallet is not maintained correctly which is difficult to assure in practice.

In this thesis, we show how to guarantee the security of a BIP32 wallet scheme without relying on the assumption of incorruptible cold wallets. To this end,

[3]This section was largely taken verbatim from [66] with slight adjustments and extensions.

we consider thresholdizing non-hardened nodes, s.t. each node consists of several devices where each of them stores a *share* of the signing secret key. This design choice allows to guarantee security even if a subset of the devices is corrupted. At a high level, our idea is to instantiate non-hardened wallets with a (t, n)-threshold signature scheme such that each non-hardened wallet is "split" into n different devices, each of which stores only a share of the signing secret key. At least $t+1$ devices are then required to sign a message. Simultaneously, the secret key of the non-hardened wallet remains secure as long as at most t devices are corrupted. All n devices store the public key and state, s.t. a single device can derive a non-hardened child public key without having to interact with the remaining $n-1$ devices. The main technical difficulties of such a thresholdized variant of the BIP32 standard are that (1) the design of threshold cryptosystems is typically rather complex and requires a careful security analysis, and (2) it is not clear how the (non-)hardened key derivation functions according to BIP32 can be translated to the threshold setting.

Threshold Signature Schemes with Rerandomizable Keys

As explained in Section 3.1.1, we showed in our formal security analysis of the BIP32 standard [67] that one can generically construct hierarchical deterministic wallets from signature schemes with rerandomizable keys. In the threshold setting that we consider in this section, we therefore require a *threshold* signature scheme with rerandomizable keys. To this end, we first provide a game-based definition of such a primitive, and then show an instantiation based on the threshold ECDSA scheme of Gennaro and Goldfeder [100]. We intentionally choose this scheme for the following two reasons: (1) it is a relatively simple scheme, i.e., it does not include advanced features such as offline signing or proactive/adaptive security which significantly increase the complexity of other threshold ECDSA schemes; (2) several threshold ECDSA schemes directly build upon the protocol of Gennaro and Goldfeder [47],[69],[48],[49], improving either its efficiency, functionality, or security. Since the general idea of these schemes is similar to the original scheme of Gennaro and Goldfeder, we believe that our results can be extended to these schemes as well.

Model of Interactive Threshold Signature Schemes with Rerandomizable Keys. An interactive (t, n)-threshold signature scheme TSig is executed among n parties $\{P_1, \cdots, P_n\}$ and consists of procedures $\mathsf{TSig} = (\mathsf{Gen}, \mathsf{TSign}, \mathsf{Verify})$, where the key generation algorithm Gen generates a public key pk and n secret key shares $\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}$ such that each party P_i learns sk_i for $i \in [n]$. The interactive signing procedure TSign is executed by all n parties, where each party takes as input its secret key share and a message, and the procedure outputs a signature for the message if at least $t + 1$ parties execute the procedure honestly. The resulting signature can then be verified via the Verify algorithm. Such a signature scheme must satisfy a notion of unforgeability, which at a high level guarantees that even if an adversary corrupts up to t parties, it cannot forge a signature.

An interactive (t, n)-threshold signature scheme with rerandomizable keys RTSig extends the above notion by two algorithms RandSK and RandPK which allow to deterministically derive rerandomized secret key shares and a rerandomized public key respectively. In more detail, the RandSK algorithm allows to rerandomize individual secret key shares, i.e., it takes as input a secret key share sk_i and a randomness ρ and outputs a rerandomized secret key share sk'_i. Similarly, the algorithm RandPK takes as input a public key pk and a randomness ρ and outputs a rerandomized public key pk'. We say that such a scheme is correct if for a set of keys $(\mathsf{pk}, \{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\})$ as generated by the Gen algorithm the following holds: the set of rerandomized secret key shares $\{\mathsf{sk}'_1, \cdots, \mathsf{sk}'_n\}$ and the rerandomized public key pk' form a valid signing key set, where each sk'_i is derived via the RandSK algorithm on input sk_i and randomness ρ, and pk' is derived via the RandPK algorithm on input pk and ρ.

We require an RTSig scheme to satisfy the property of rerandomizability of public keys, which guarantees that a rerandomized public key is computationally indistinguishable from a freshly generated public key. This is a slightly weaker notion than the perfect rerandomizability of keys of rerandomizable signature schemes (cf. Definition 2.3.1) which requires rerandomized public and secret keys to be *identically* distributed to a freshly generated key pair. We note, however, that this weaker rerandomizability property is sufficient for the wallet setting, where only public keys must be unlinkable, i.e., computationally indistinguishable from freshly generated public keys.

Finally, we require an RTSig scheme to satisfy the security notion of *one-per message unforgeability of interactive threshold signature schemes with honestly rerandomizable keys* which we formally define via a security game denoted by **th−ufcma−hrk1**. This notion essentially combines the two notions of one-per message unforgeability of signature schemes with honestly rerandomizable keys (cf. Definition 2.3.3) and unforgeability of interactive threshold signature schemes. That is, at the beginning of game **th−ufcma−hrk1** the adversary corrupts up to t parties and consequently obtains those parties' secret key shares and the scheme's public key. The adversary and the challenger, playing the role of the honest parties, then jointly execute signing procedures for messages chosen by the adversary and for honestly rerandomized keys with the restriction that a message can only be signed once for one honestly rerandomized key set. Finally, the adversary wins the game if it outputs a valid forgery under an honestly rerandomized public key and for a message that was never signed previously under this rerandomized public key.

Interactive Threshold ECDSA with Rerandomizable Keys. We show how to extend the interactive threshold ECDSA scheme as proposed by Gennaro and Goldfeder [100] (with slight adjustments) to an interactive threshold ECDSA scheme with rerandomizable keys (which we denote by rGG) by providing RandSK and RandPK algorithms for the rerandomization of secret key shares and the public key respectively. Recall that the ECDSA signature scheme is defined w.r.t. a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q and that an ECDSA key pair $(\mathsf{pk}, \mathsf{sk})$ is simply computed as $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow \mathsf{sk} \cdot G$. In the scheme of Gennaro and Goldfeder, each party P_i receives a secret key share sk_i of the full secret key sk such that all $\mathsf{sk}_1, \cdots, \mathsf{sk}_n$ lie on a degree-t polynomial with free term sk. Each party additionally holds a public key share $\mathsf{sk}_i \cdot G$. The main technical challenge when extending the scheme of Gennaro and Goldfeder to a key rerandomizable scheme is that we must find a non-interactive algorithm RandSK, which rerandomizes a secret key share with a randomness ρ such that the following properties hold: (1) all secret key shares rerandomized with ρ form a valid sharing of the rerandomized secret key $\mathsf{sk} + \rho \mod q$, and (2) RandSK must be deterministic. At a high level, our RandSK algorithm, on input a secret key share sk_i and a randomness ρ, deterministically generates a degree-t polynomial F, which shares ρ, and computes the rerandomized secret key share as $\mathsf{sk}'_i \leftarrow \mathsf{sk}_i + F(i) \mod q$. More concretely, RandSK generates the polynomial $F(x) \coloneqq a_t x^t + \cdots + a_1 x + \rho$ where the coefficients are derived as $a_k \leftarrow \mathsf{H}(\rho, k)$ for $k \in [t]$ and where H denotes a cryptographic hash function. The algorithm then evaluates $F(i)$, which essentially yields a share ρ_i of randomness ρ. This randomness share is then added to sk_i to compute the rerandomized secret key share $\mathsf{sk}'_i \leftarrow \mathsf{sk}_i + \rho_i \mod q$. That is, sk'_i is essentially a share of the secret key $\mathsf{sk} + \rho \mod q$. The RandPK algorithm works correspondingly for the public key and public key shares. Note that it is crucial that $F(x)$ is computed deterministically to ensure that all parties individually compute the same polynomial for the same randomness ρ.

We provide a proof sketch that shows that our construction rGG satisfies the property of rerandomizability of public keys. Essentially, we show that when H is modeled as a random oracle [24], we can make a reduction to the discrete logarithm problem to show that rGG satisfies the property of rerandomizability of public keys. We additionally show a proof sketch that our rGG scheme satisfies one-more unforgeability under honestly rerandomizable keys. Essentially, we can show a reduction to the one-per message unforgeability of the ECDSA scheme with additively rerandomizable keys that we presented in Section 3.1.1 and in the publication [67] to prove the one-per message unforgeability of our rGG scheme.

BIP32-Compatible Threshold Wallets

With our additively rerandomizable threshold ECDSA scheme rGG in place, we will discuss in the following how the respective wallet derivations of a BIP32 wallet can be implemented in the threshold setting. In particular, we consider the following setting for our threshold BIP32 wallet: All non-hardened wallets are thresholdized, i.e., each non-hardened wallet consists of n devices which execute a (t, n)-threshold signature scheme with rerandomizable keys. Hardened wallets, on the other hand, are single devices (i.e. not thresholdized), since the corruption of a hardened wallet does not affect the security of the remaining wallets in the tree. We do not allow hardened wallets to derive child wallets, i.e., hardened wallets always represent leafs in the wallet tree. Therefore, we assume that in both cases, i.e., the non-hardened and hardened wallet derivation, the parent wallet is non-hardened and thresholdized. In the following, we explain how the (non-)hardened derivation mechanisms can be translated to the threshold setting w.r.t. to our additively rerandomizable threshold ECDSA scheme.

Non-Hardened Derivation The derivation of non-hardened nodes in the threshold setting is fairly straightforward and follows the ideas of the BIP32 standard. Essentially, a non-hardened parent node identified by ID and consisting of n devices s.t. each device stores a secret key share $\mathsf{sk}_{i,\mathsf{ID}}$ and the state $\mathsf{St}_{\mathsf{ID}}$ can derive a thresholdized non-hardened child wallet as follows: First, each device of the parent node computes locally $(\rho, \mathsf{St}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}')$ and $\mathsf{sk}_{i,\mathsf{ID}'} \leftarrow \mathsf{rGG}[\mathsf{H}_0].\mathsf{RandSK}(i, \mathsf{sk}_{i,\mathsf{ID}}, \rho)$. Then the devices of the parent node must forward the rerandomized secret key shares $\mathsf{sk}_{i,\mathsf{ID}'}$ and the state $\mathsf{St}_{\mathsf{ID}'}$ to the n devices of the child node. The forwarding of the state $\mathsf{St}_{\mathsf{ID}'}$ is straightforward, since we assume an honest majority among the parent devices and since each parent device knows $\mathsf{St}_{\mathsf{ID}'}$. That is, all parent devices can simply send $\mathsf{St}_{\mathsf{ID}'}$ to all child devices. Each child device then receives at least $t + 1$ times the value $\mathsf{St}_{\mathsf{ID}'}$ which it uses as the node's state. The forwarding of the secret key shares $\mathsf{sk}_{i,\mathsf{ID}'}$ is more involved and requires a protocol involving $2n$ devices (n child and n parent wallet devices) of which a total of $2t$ devices can be corrupted. Note that a simple forwarding of secret key share $\mathsf{sk}_{i,\mathsf{ID}'}$ to the i-th device of the child wallet is insecure as it allows an adversary to learn a total of $2t$ secret key shares. Instead, the $2n$ devices must engage in the execution of a dynamic proactive secret sharing (DPSS) scheme (e.g., [20, 138, 161]), which allows to *securely* handover the rerandomized key shares to the devices of the child node even in the presence of $2t$ corrupted devices. Note that DPSS schemes typically incur a significant communication overhead since all $2n$ parties must interact with each other.

Hardened Derivation The main challenge when considering BIP32 wallets in the threshold setting is the hardened node derivation mechanism. Recall that the derivation of a hardened node according to BIP32 requires the computation of $(\rho, \mathsf{St}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}')$, i.e., the evaluation of a hash function where one of the inputs is the parent secret key. In the threshold setting, however, the secret key $\mathsf{sk}_{\mathsf{ID}}$ is shared among n devices such that no single device knows the full key. It is therefore not at all clear how $\mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{St}_{\mathsf{ID}}, \mathsf{ID}')$ can be computed efficiently without naively reconstructing $\mathsf{sk}_{\mathsf{ID}}$ (which would trivially break the security of the wallet). Furthermore, in the hardened derivation, each parent device can only learn a randomness *share* ρ_i instead of the entire randomness ρ. To see why that is the case, consider the setting where an adversary corrupts the hardened node, thereby learning its secret key $\mathsf{sk}_{\mathsf{ID}} + \rho$, as well as a parent node device, thereby learning ρ. The adversary could then trivially learn the parent node's secret key.

One obvious (and to the best of our knowledge only) way to resolve the above issues is using generic multi-party computation (MPC) techniques [53, 105, 106], which allow to securely compute any function in a distributed setting without revealing the function inputs. However, generic MPC is inherently inefficient, in particular since the BIP32 standard uses the well-known hash function SHA-512, which is known to be only inefficiently computable via MPC [38].

Due to this limitation, we consider a more efficient hardened node derivation mechanism, which achieves the same properties as the one originally specified in BIP32. At a high level, instead of having the parent wallet devices rerandomize their secret key shares and forward them to the hardened wallet, we simply let the parent devices generate a random value from which the hardened node can deterministically derive its own keys. For the computation of the random seed, we employ the threshold verifiable random function (TVRF) from Galindo et al. [97]. A (t, n)-TVRF is a cryptographic primitive that consists of four algorithms, namely $(\mathsf{Gen}, \mathsf{PEval}, \mathsf{Combine}, \mathsf{Verify})$, and is executed by n parties. The Gen algorithm outputs a secret key share sk_i to each party P_i as well as a public key pk, and each party can use their secret key share to deterministically compute an evaluation share ϕ_i and proof π_i on a message m using algorithm PEval. Given at least $t + 1$ valid evaluation shares for m, any party can deterministically compute a pseudorandom value ϕ and a proof π using the $\mathsf{Combine}$ algorithm. Finally, given the public key pk, the value ϕ and the proof π, any party can verify that ϕ

was computed correctly. A TVRF satisfies three security properties, namely *pseudorandomness*, *uniqueness*, and *robustness*: pseudorandomness guarantees that ϕ is a pseudorandom value, uniqueness guarantees that for a unique message m the TVRF outputs a unique ϕ, and robustness guarantees that if the proof π verifies w.r.t. pk and ϕ, then the value for ϕ was computed correctly.

We use the TVRF for the hardened wallet derivation in the following way: Each device of the non-hardened parent node maintains a secret key share for the TVRF and, upon the derivation of a hardened node with identifier ID, uses this secret key share to compute an evaluation share ϕ_i and the corresponding proof π_i on ID. It then sends (ϕ_i, π_i) to the hardened node, which combines $t+1$ shares to a pseudorandom seed ϕ and verifies the correctness of ϕ using the proof π and the public key of the TVRF. Note that any set of $t+1$ correct evaluation shares will yield the same seed ϕ, but only including a single invalid evaluation share will lead to a different seed. Therefore, the verifiability of the final seed is crucial to our solution. We use the TVRF from [97], which we denote by TVRF and which is not only deterministic and one-way but also non-interactively computable, therefore exhibiting the same properties as the original BIP32 derivation mechanism. We present our improved hardened node derivation mechanism w.r.t. TVRF pictorially in Figure 3.1.

While the above solution is compatible with BIP32 (since it achieves the same properties), it has the significant drawback that each non-hardened device must maintain two secret key shares, one for the signature scheme rGG and one for the TVRF scheme. As a consequence, each device requires double the storage space which is an issue for space restricted devices. Another even more severe issue, however, is that similar to the signing keys the TVRF keys must be deterministically derived throughout the wallet tree via executions of a communication heavy DPSS scheme. This incurs a significant communication overhead, especially since *all* non-hardened nodes must derive TVRF keys irrespectively of whether they want to derive a hardened node or not.

To this end, we make the following observation: the DDH-based TVRF scheme of [97] and the ECDSA signature scheme both operate over a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q and use secret/public key pairs $\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow \mathsf{sk} \cdot G$. The security of TVRF relies on the assumption that DDH is hard in $\mathbb{G}$. Bitcoin, Ethereum and several other cryptocurrencies use the group $\mathbb{G}$ identified by the elliptic curve secp256k1, for which dlog and DDH are assumed to be hard. Therefore, our idea to mitigate the above issues is to use only a single key pair for both schemes. This allows non-hardened wallets to re-use their signing secret key shares from scheme rGG for the TVRF scheme during the hardened node derivation, thereby avoiding the overhead of maintaining a second key pair per wallet instance.

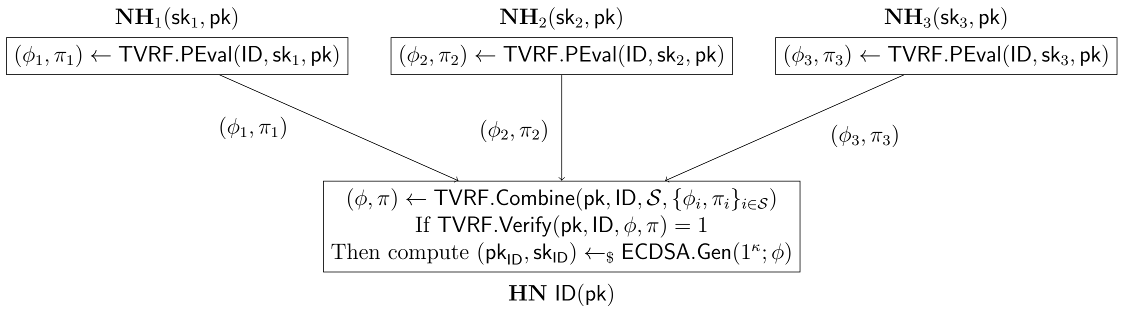

$\mathbf{NH}_1(\mathsf{sk}_1, \mathsf{pk})$
$(\phi_1, \pi_1) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_1, \mathsf{pk})$
$\mathbf{NH}_2(\mathsf{sk}_2, \mathsf{pk})$
$(\phi_2, \pi_2) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_2, \mathsf{pk})$
$\mathbf{NH}_3(\mathsf{sk}_3, \mathsf{pk})$
$(\phi_3, \pi_3) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_3, \mathsf{pk})$
(ϕ_1, π_1)
(ϕ_2, π_2)
(ϕ_3, π_3)
$(\phi, \pi) \leftarrow \mathsf{TVRF.Combine}(\mathsf{pk}, \mathsf{ID}, \mathcal{S}, \{\phi_i, \pi_i\}_{i \in \mathcal{S}})$
If $\mathsf{TVRF.Verify}(\mathsf{pk}, \mathsf{ID}, \phi, \pi) = 1$
Then compute $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}) \leftarrow_{\$} \mathsf{ECDSA.Gen}(1^\kappa; \phi)$
$\mathbf{HN}$ $\mathsf{ID}(\mathsf{pk})$

Figure 3.1.: Pictorial representation of our improved hardened node derivation mechanism in the threshold setting. Each of the three devices $\mathbf{NH}_1$, $\mathbf{NH}_2$, $\mathbf{NH}_3$ of the non-hardened parent node stores a TVRF public key pk and secret key share sk_i for $i \in [3]$. In order to derive a hardened node $\mathbf{HN}$ with identity ID, each non-hardened device locally evaluates the TVRF on input ID and sends the resulting evaluation share to $\mathbf{HN}$. The hardened node can then choose a subset $\mathcal{S}$ of the set [3], combine the corresponding evaluation shares to a full random value ϕ, verify that the non-hardened devices in $\mathcal{S}$ behaved honestly, and then use ϕ as input to the key generation algorithm of the ECDSA signature scheme. Note that this key generation is deterministic, since we explicitly give the randomness ϕ as input. Figure taken from [66].

While it is typically not recommended to re-use a cryptographic key pair for several primitives, we show that it is indeed secure to use the same key pair for the rGG and the TVRF scheme. In order to do so, we first define a joint scheme which essentially consists of the combined procedures of rGG and TVRF except that it only uses one of the respective key generation algorithms. We then formally define the security properties *pseudorandomness*, *uniqueness*, and *robustness* for the joint scheme. These security notions essentially combine the respective security properties of the TVRF scheme with the one-more unforgeability notion of our rGG scheme. That is, for each of the above security notions, we define a game, where an adversary (1) can corrupt t parties, (2) receives oracle access to all oracles of the one-more unforgeability game (i.e., $\mathbf{th{-}ufcma{-}hrk1}$), and all oracles of the respective TVRF property (e.g., pseudorandomness), and (3) can win the game by either breaking the one-more unforgeability of rGG or the TVRF property.

We finally show that the joint scheme indeed satisfies our security properties. For instance, in order to show that the joint scheme satisfies our pseudorandomness property, we provide a reduction from the $\mathbf{uf{-}cma{-}hrk1}$-security of the rGG scheme and from the pseudorandomness of the TVRF scheme. That is, we essentially show that if there exists an adversary that can break the pseudorandomness property of our joint scheme, then we can construct an adversary that can either break the **uf−cma−hrk1**-security of rGG or the pseudorandomness of TVRF. The difficulty in this reduction is that the reduction does not know in advance how the adversary against our joint scheme is going to break the pseudorandomness property. That is, the reduction has to guess in advance whether to reduce to the **uf−cma−hrk1**-security of rGG or to the pseudorandomness of TVRF. However, in case of a reduction to the pseudorandomness of TVRF the adversary against the joint scheme is allowed to receive signatures under the scheme's secret key, while the reduction does not obtain access to a signing oracle. The reduction therefore must simulate the signing protocol to the adversary in the joint scheme without having access to a signing oracle itself. Since our reduction guesses in advance how the adversary against our joint scheme is going to break the pseudorandomness property, the reduction incurs a loss of $\frac{1}{2}$.

3.2. Related Work

In this section, we review related works on the topics of cryptographic wallets and threshold ECDSA schemes.

3.2.1. Cryptographic Wallets

(Hierarchical) Deterministic Wallets.[4] Cryptographic wallets have been extensively studied in the past. In particular the introduction of the BIP32 standard in 2012 [179] and its wide use in practice has sparked many academic works analyzing the notion of hierarchical deterministic wallets. Gutoski and Stebila [111] presented a hierarchical deterministic wallet scheme that deviates from the BIP32 standard. Their scheme remains secure if at most m child keys are leaked, where m is a parameter that is fixed upon setup of the wallet. The authors prove their scheme secure under the *one-more discrete log* assumption [21], however in a weak security model. Later, Das et al. [68] gave the first formal analysis of deterministic wallets in the hot/cold setting, and provided a construction based on multiplicatively rerandomizable ECDSA. However, in contrast to our publication [67], Das et al. did not consider the hierarchical setting, where child nodes can derive further children. Alkadri et al. [5] translated the model of deterministic wallets in the hot/cold setting by Das et al. [68] to the post-quantum setting (cf. Chapter 4).

[4]Parts of this paragraph were taken verbatim from [66] with some adjustments.

Essentially, Alkadri et al. introduced the notion of a post-quantum secure deterministic wallet in the hot/cold setting, provided a generic construction of such a wallet scheme from any post-quantum secure signature scheme with rerandomizable keys, and showed a concrete instantiation. Hu [114] later followed-up on the work of Alkadri et al. by providing an instantiation of a post-quantum secure signature scheme with rerandomizable keys that allows for smaller public key and signature sizes than the rerandomizable signature scheme proposed by Alkadri et al. Fan et al. [89] introduced a hierarchical deterministic wallet scheme for Schnorr signatures [160] that is based on trapdoor hash functions. However, their work lacks a formal security model and analysis. Luzio et al. [132] presented a hierarchical deterministic wallet scheme, which relies on hierarchical key assignment schemes [12] and allows to leak child secret keys without compromising the security of the entire wallet hierarchy. However, their scheme is not compatible with Bitcoin as it requires a more complex signature verification algorithm. Similarly, Yin et al. [181] recently proposed a model for hierarchical deterministic wallets supporting stealth addresses, however, they provide a construction that is incompatible with Bitcoin as it relies on bilinear maps. Erwig and Riahi [83] recently proposed deterministic wallets with support for adaptor signatures (cf. Chapter 5). In a recent white paper [180], Lindell described the design of a shared-custodial wallet that is being developed and used by the cryptocurrency exchange company Coinbase[5]. At a high level, the shared-custodial wallet shares the signing secret key between a user and a service provider (namely Coinbase), and it supports hierarchical deterministic key derivation mechanisms corresponding to BIP32. That is, the setting considered in the white paper [180] is similar to the setting of our work on BIP32-compatible threshold wallets [66]. Indeed, similar to our work, Lindell described that the hardened key derivation as specified by BIP32 is not suitable for a setting where the signing key is shared among several parties. Instead, Lindell devised an alternative hardened key derivation that deviates from the original BIP32 specification and that makes use of a threshold verifiable random function. While this is similar to our work [66], the white paper crucially differs from our work as follows: (1) the white paper lacks a formal description and security analysis of the proposed solution, and (2) the white paper does not include the key re-use mechanism which is one main contribution of our work. Finally, Chuang et al. [58] recently studied BIP32 as is in the two-party setting. In particular, the authors considered the exact hardened key derivation mechanism as specified by BIP32 translated to the two-party setting where the signing secret key is shared between two parties. The authors provided an implementation and evaluated the

[5]`https://www.coinbase.com/`

efficiency of the hardened derivation in the two-party setting.

Hardware Wallets. Another line of research focused on hardware wallets. Arapinis et al. [11] provided a formal security model of hardware wallets in the UC framework [46]. Marcedone et al. [139] considered the scenario of a user who knows a low-entropy password and uses a hardware wallet, which stores a high-entropy secret, to generate signatures. To this end, they essentially provided a generic construction of a two-party signature scheme which is executed between the user and the hardware wallet. The construction can be instantiated with either Schnorr or ECDSA. However, their scheme is prone to offline password-guessing attacks, i.e., a corrupted hardware wallet can brute-force the user's password and thereby forge signatures. Gentilal et al. [102] implemented a cryptographic wallet in the hot/cold setting where the cold wallet is executed in a trusted execution environment to further prevent it from corruption. Bamert et al. [18] proposed *BlueWallet*, a hardware wallet that stores the user's secret key and connects via Bluetooth to an untrusted user device to generate signatures on behalf of the user.

Other Related Works. Several works [41, 42, 62, 148] investigated the consequences that implementation errors or the sampling of weak randomness can have on the security of wallets. Makriyannis and Peled [149] and Makriyannis and Yomtov [148] respectively found attacks on two popular threshold ECDSA schemes [100, 101], which are widely used in practice. However, all of the described attacks can be mitigated by appropriately adjusting the affected schemes. Turuani et al. [177] analyzed the Electrum wallet using automatic analysis, and showed that it is secure in the Dolev-Yao model.

Kondi et al. [123] proposed a threshold wallet solution that allows to proactively refresh the scheme's secret key even while a subset of the protocol participants are offline. The offline parties can then non-interactively update their respective secret key share once they come back online. The authors show constructions for threshold ECDSA and Schnorr in the $(2, n)$-setting, where two parties are required to compute a signature. Additionally, the authors show that such a proactive refresh with offline parties is impossible in the (t, n)-setting in case the number of online parties is below a certain threshold.

Zindros [184] introduced a keyless wallet based on witness encryption [99] and smart contracts, where a user can spend its funds using only a password. Chaum et al. [55, 56] introduced a backup mechanism that allows a wallet user to prove that it is the owner of the wallet's secret key, even when the secret key is leaked. Eyal [88] analyzed the general design of secure wallets. That is, the author analyzed whether and how much the security of a wallet is dependent on various factors such as the number of keys it maintains, the combination of keys required for the generation of a valid signature, or the implementation type (hardware and/or software) of a wallet. Finally, Mangipudi et al. [137] provided a user study on the topic of multi-device wallets and very recently Zyskind et al. [185] proposed the notion of unstoppable wallets, which execute a threshold ECDSA signature scheme between a set of parties and a smart contract that runs on the blockchain.

3.2.2. Rerandomizable Signatures and Threshold ECDSA

Rerandomizable Signatures. The notion of signature schemes with rerandomizable keys has first been introduced by Fleischhacker et al. [94], who also showed an instantiation from the Schnorr signature scheme [160]. Since then this notion has been shown to be a useful building block for deterministic wallet schemes (e.g., [5, 67, 68]). In particular, Das et al. [68] proved the security of a multiplicatively rerandomizable ECDSA scheme, while we presented and proved secure an additively rerandomizable ECDSA scheme in this chapter of the thesis (cf. Section 3.1.1). In a follow-up work, Groth and Shoup [109] presented a formal security analysis of an additively rerandomizable ECDSA scheme that supports the computation of pre-signatures, i.e., it allows to pre-process a part of the signature even before the message, that is to be signed, is known. While their scheme achieves a slightly stronger security than the one we analyzed in this chapter (their security model does not restrict the adversary to obtain at most one signature per message and derived public key), their analysis relies on the idealized generic group model [147, 164]. Our analysis, however, is only based on the random oracle model [24].

Our additively rerandomizable ECDSA scheme uses public key prefixing which means that in order to generate a signature for a message m under key pair $(\mathsf{pk}, \mathsf{sk})$, the message must be prefixed with pk. That is, the message that is being signed is the concatenation of pk and m. This public key prefixing is crucial for our security proof. A recent work by Hanzlik et al. [113] shows how to avoid such public key prefixing by instead prefixing signed messages with an index.

Finally, the concept of related key attacks has previously been shown to be a useful proof technique for signature schemes with rerandomizable keys [68, 94].

Threshold ECDSA.[6] In recent years, there has been a huge interest in threshold ECDSA schemes (e.g., [3, 30, 47, 48, 49, 64, 69, 129, 130]) motivated by their use in distributed cryptographic wallets. For a more in-depth comparison of different

[6]This paragraph was largely taken verbatim from [66] with some adjustments.

threshold ECDSA schemes, we refer to the survey of Aumasson et al. [14]. In this chapter, we base our solution for BIP32-compatible threshold wallets [66] on the threshold ECDSA scheme of Gennaro and Goldfeder [100]. Similarly to our work in this chapter, a recent work by Groth and Shoup [108] introduced a threshold ECDSA scheme with additive key rerandomization according to the BIP32 specification. However, the authors did not consider the derivation of hardened nodes in the threshold setting, which is the main focus of our work [66]. Groth and Shoup analyzed their scheme in the ideal/real world setting w.r.t. an ECDSA-specifc functionality, whereas we give a general game-based definition for threshold signature schemes with rerandomizable keys, and show that the construction of Gennaro and Goldfeder [100] can be extended to satisfy our definition. Finally, the scheme of Groth and Shoup is rather complex, whereas we were aiming for a simple threshold ECDSA scheme with rerandomizable keys for our BIP32-compatible threshold wallet construction.

4. Deterministic Wallets with Post-Quantum Security

Most major cryptocurrencies nowadays, including Bitcoin [146] and Ethereum, use digital signature schemes that are not secure against quantum adversaries, i.e., adversaries with access to quantum computing power. More concretely, the security of the two most widely used signature schemes in the cryptocurrency space, namely ECDSA and Schnorr, is based on the hardness assumption of computing discrete logarithms. However, in 1994 Shor introduced an algorithm [163] which can efficiently compute discrete logarithms on a quantum computer and thereby break many schemes that are presumed to be secure in the classical (non-quantum) setting. Recall that a public/secret key pair of the ECDSA signature scheme (and likewise of the Schnorr signature scheme) has the form $(\mathsf{pk}, \mathsf{sk}) := (x \cdot G, x)$ for a uniform random $x \leftarrow_{\$} \mathbb{Z}_q$ and for a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q. Then it is easy to see that if a quantum adversary, which knows pk and which can efficiently compute discrete logarithms in group $\mathbb{G}$ using Shor's algorithm, can compute sk and thereby trivially forge signatures. Naturally, this would render a cryptocurrency network that is based on the ECDSA or Schnorr signature scheme insecure as no user would be able to securely store and spend its funds.

The field of post-quantum cryptography investigates cryptographic primitives that are secure against both, classical and quantum adversaries, but that can be executed by classical devices. This is a particularly interesting approach as it offers protection against quantum adversaries already in today's world, where quantum computers are not yet sufficiently developed for mass adaption. A compelling idea to protect cryptocurrency networks from quantum adversaries might therefore be to use only post-quantum secure cryptographic building blocks in the network. Indeed, several works (e.g., [4, 28, 60, 86, 87, 155]) have explored the direction of post-quantum secure cryptocurrencies in the past years. For instance, the "Bitcoin Post-Quantum" [28] and the "Quantum Resistant Ledger" [155] projects use post-quantum secure hash-based signature schemes, whereas Esgin et al. [86, 87] explore privacy-preserving cryptocurrency networks that use only post-quantum secure cryptographic building blocks.

4.1. Our Contribution

In this thesis, we advance the studies on post-quantum secure cryptocurrency networks by providing the first deterministic wallet construction that is resistant to quantum adversaries. This chapter is based on the following publication which can be found (with some adjustments[1]) in Appendix C:

[5] N. A. Alkadri, P. Das, A. Erwig, S. Faust, J. Krämer, S. Riahi, and P. Struck. "Deterministic Wallets in a Quantum World". In: *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.* 2020, pp. 1017–1031. **Part of this thesis.**

More concretely, in the above publication, we (1) consider the model of deterministic wallets in the hot/cold setting as introduced by Das et al. [68] in the post-quantum setting; (2) give a generic construction of post-quantum secure signature schemes with rerandomizable public keys from lattice-based Fiat-Shamir signature schemes; (3) give a generic construction of post-quantum secure deterministic wallets from signature schemes with rerandomizable public keys; and (4) provide an evaluation of our results by assessing the potential transaction throughput that a cryptocurrency network could achieve if it uses our signature scheme with rerandomizable public keys. In the following we describe our contribution in more detail.

4.1.1. Post-Quantum Secure Deterministic Wallets

In this subsection, we first recall the post-quantum security model, and subsequently we describe our generic post-quantum secure deterministic wallet scheme.

The Post-Quantum Security Model The post-quantum security model assumes that an adversary that has access to a quantum computer, whereas honest parties run on classical machines only. Defining a security game in this setting requires some additional care as compared to the classical setting: While the adversary in the classical setting can query oracles only classically, this is not necessarily the case in the post-quantum setting. Here, we must distinguish whether the adversary can query an oracle classically or whether it can access it using its quantum computing power and thereby query it in superposition. Essentially, oracles that model some form of computation by honest parties, which run on classical machines, can only

[1]We consider a slightly adjusted version of publication [5] in this chapter, which fixes a small mistake contained in the original publication. We provide the adjusted version and further details in Appendix C.

be queried classically by a quantum adversary. However, other oracles, such as a random oracle, model public functions that in practice can be implemented and evaluated by the adversary on its quantum computer. Therefore, the security model must allow the adversary to query such oracles in superposition. Indeed, for this reason Boneh et al. [34] introduced the quantum random oracle model (QROM), where the adversary receives access to a random oracle that it can query in superposition.

We note that this post-quantum model best reflects the real world as it assumes that the cryptographic scheme can be run on classical machines while it offers security against quantum adversaries. A stronger but less relevant security model is the so-called fully-quantum model, where even honest parties are assumed to run on quantum computers. In this thesis, we consider only the post-quantum model.

Post-Quantum Secure Signature Schemes with Rerandomizable Keys. As a first step towards our post-quantum secure deterministic wallets, we present for the first time a construction of a signature scheme with rerandomizable keys in the post-quantum setting. While there are several promising approaches to construct post-quantum secure cryptographic schemes, we base our construction on the computationally hard lattice problem known as module learning with errors [39, 125] (MLWE). This problem has the advantages that (1) the MLWE assumption is well studied, which increases the confidence in its conjectured security, and (2) its linear structure allows for a simple key rerandomization mechanism. We give a construction of a signature scheme with rerandomizable keys based on a generic MLWE-based Fiat-Shamir signature scheme. Our scheme resembles the one of classical Schnorr signatures but in the lattice setting. In particular, the public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ is an instance of the MLWE problem, where sk is distributed according to a Gaussian distribution over some set. We rerandomize sk with a randomness ρ by first deterministically sampling a Gaussian distributed secret key sk' from ρ and then adding sk' to sk. The resulting rerandomized secret key is again Gaussian distributed but with a different standard deviation than sk. That is, in our construction rerandomized secret keys are distinguishable from freshly generated secret keys. Due to this, we formalize the notion of signature schemes with rerandomizable public keys, which requires rerandomized public keys to be computationally indistinguishable from freshly generated public keys, but which does not guarantee the same for secret keys. This is similar to our notion of interactive threshold signature schemes with rerandomizable keys (cf. Chapter 3) and we again argue that for the wallet setting this notion of public key rerandomizability suffices. We additionally require that signature schemes with rerandomizable public keys satisfy a simulatability property, which at a high level guarantees the following: there must exist an efficient algorithm which on input a public key pk and a message m outputs a signature which is computationally indistinguishable from a correctly generated signature for pk and m (i.e., using secret key sk corresponding to pk). Intuitively, we require this simulatability notion for the security proof of our generic wallet construction, where we have to simulate signing oracle queries to the adversary without knowing the scheme's secret key.

Post-Quantum Secure Deterministic Wallets. As a next step, we describe a model for post-quantum secure deterministic wallets, and we show a generic construction from any post-quantum secure signature scheme with rerandomizable public keys. The model is essentially a post-quantum variant of the model of Das et al. [68] for deterministic wallets in the hot/cold setting. That is, as compared to the full hierarchical deterministic wallet model as presented in Chapter 3, our model allows only the master wallet to derive children, and all wallet instances are assumed to be implemented in the hot/cold setting. The security properties of wallet unlinkability and wallet unforgeability are defined in the same way as in the model of Das et al. [68] with the difference that our model considers a quantum adversary that can access the random oracle in superposition, i.e., it receives access to a quantum random oracle.

With the model in place, we then show that we can generically construct post-quantum secure deterministic wallets in the hot/cold setting from any post-quantum secure signature scheme with rerandomizable public keys, and we prove that our construction satisfies the notions of wallet unlinkability and wallet unforgeability in the post-quantum setting. The main difficulty in these proofs is that the quantum adversary can query the random oracle in superposition, whereas the challenger is only classical. It is therefore not immediately clear how the challenger can simulate the quantum random oracle to the adversary. At a high level, we deal with this issue by using two established techniques, namely the one-way to hiding lemma [10, 178] and Zhandry's small-range distributions [182].

4.1.2. Evaluation of our Results

Finally, we discuss how to instantiate our signature scheme with rerandomizable keys with the lattice-based signature scheme qTesla [6] and, given this concrete instantiation, we provide an overview of the transaction throughput that a blockchain network could achieve with our scheme. In order to do so, we estimate the size of a typical Bitcoin transaction where the ECDSA signature scheme is replaced by qTesla. Recall that a simple Bitcoin transaction, i.e., a transaction with only a single sender input and receiver output, contains the public key and a signature of the sender. The transaction contains additionally the hash of the public key of the receiver, which has the same size for the ECDSA and qTesla scheme. In our work, we estimate the size of a raw Bitcoin transaction, i.e., a transaction without sender public key and signature, to be around 100 bytes. The size of a qTesla public key and signature is $14,880$ bytes and $2,592$ bytes respectively [6]. Consequently, a Bitcoin transaction where ECDSA is replaced by qTesla has the size: 100 bytes $+14,880$ bytes $+2,592$ bytes $= 17,572$ bytes. As a comparison, we estimate the size of the same transaction using ECDSA instead of qTesla to be roughly 240 bytes[2]. Unsurprisingly, the size of a Bitcoin transaction using qTesla is significantly bigger than when the same transaction uses the classically secure ECDSA signature scheme. This directly impacts the transaction throughput, i.e., less transactions can be processed per second in the post-quantum setting compared to the classical setting. While there exist general techniques to increase the transaction throughput of a cryptocurrency such as increasing the block size or decreasing the time required until a block of transactions is considered final, these solutions do not fundamentally address the transaction throughput issue and come with various disadvantages. A more promising approach to increase the transaction throughput in the post-quantum setting is to find a post-quantum secure signature scheme with smaller public key and signature sizes. We regard our work as a first step towards post-quantum secure deterministic wallets, which can be improved upon in future works. Indeed, Hu [114] recently presented a post-quantum secure signature scheme with rerandomizable keys with reduced public key and signature sizes compared to our post-quantum secure rerandomizable signature scheme. The scheme by Hu is based on Falcon [95], a lattice-based signature scheme that follows the hash-then-sign paradigm [103].

4.2. Related Work

We refer to Section 3.2.1 for a comprehensive overview over related works on the topic of cryptographic wallets. Here, we focus on related works w.r.t. (post-) quantum secure blockchain networks as well as lattice-based Fiat-Shamir signatures.

Many works have proposed cryptographic primitives with security against quantum adversaries that can be used in blockchain networks (e.g. [8, 9, 85, 183]). In particular, the concept of a so-called ring confidential transaction (RingCT) with

[2]In this estimate, we consider uncompressed ECDSA public keys [29] which have a size of 65 bytes. Compressed public keys, instead, have a size of 33 bytes, which would then lead to a total transaction size of roughly 208 bytes.

post-quantum security has been extensively studied (e.g., [86, 87, 98, 173, 174]). A RingCT protocol essentially consists of several cryptographic primitives such as a linkable ring signature scheme [131, 156], a zero-knowledge proof system [32], and a commitment scheme [31], and allows to guarantee strong privacy guarantees for users in a blockchain network. More concretely, when a transaction is sent to the blockchain via a RingCT protocol, the transaction hides the sender's identity as well as the amount of coins spent. The popular privacy-preserving cryptocurrency network Monero [150] uses such a RingCT scheme, albeit one that is only classically secure.

A related line of work has focused on the construction of post-quantum secure adaptor signatures [84, 104, 122, 168] which is a cryptographic primitive that has wide applications in blockchain networks. For a comprehensive overview over (post-quantum secure) adaptor signatures, we refer to Section 5.3 of this thesis. A comprehensive survey on various post-quantum secure signature schemes with advanced features that have proven to be useful in the blockchain setting was provided by Buser et al. [43].

Several works explored the (post-) quantum security of Bitcoin (e.g., [4, 60, 128, 159]) and various industry projects such as the "Bitcoin Post-Quantum" [28], the "Quantum Resistant Ledger" [155], or the "Abelian" [2] projects build post-quantum secure blockchain networks.

Finally, lattice-based Fiat-Shamir signatures have been widely studied in the past (e.g. [6, 73, 120, 134, 144]), however, no prior work considered lattice-based signature schemes with rerandomizable keys.

5. Deterministic Wallets for Adaptor Signatures[1]

Most blockchain and cryptocurrency networks, including Bitcoin, only support the execution of simple applications while others, such as Monero or Zcash, are even more restrictive in their functionality and only support simple payments [26, 158]. While the functionality of blockchains can be extended by appropriately adjusting their mining algorithms, this requires a hard fork of the blockchain code which can take several years to complete in practice. In order to improve the restricted functionality of many blockchains without having to change the blockchain implementation and to allow for the execution of a larger class of applications, a new type of signature scheme called *adaptor signatures* was introduced by the cryptocurrency community [152] and first formally defined by Aumayr et al. [15].[2] At a high level, adaptor signatures allow two parties, a *signer* and a *publisher*, to exchange a signature for a secret value. More concretely, adaptor signatures consider the scenario where the publisher knows an instance of a hard relation, i.e., a statement and witness pair and the signer holds a signing secret key. The publisher can then send the statement to the signer who, using its secret key and the statement, generates an incomplete signature called *pre-signature* which can be *adapted* by the publisher to a full valid signature using the witness. Once the adapted full signature is published on the blockchain, the signer can extract the witness given the pre- and full signature.

Adaptor signatures have proven to be extremely versatile for blockchain applications. They allow for efficient constructions of two important categories of applications, namely payment channels (e.g., [15, 167]) and atomic swaps (e.g., [70, 171]), while requiring only a minimal functionality from the underlying blockchain. Payment channels are a so-called off-chain solution, which allows two parties to issue many micropayments to each other without incurring fees for each transac-

[1]The introduction of this Chapter as well as Section 5.2.1 were largely taken verbatim from [83] with some adjustments.

[2]Fournier [96] concurrently provided a weaker model for adaptor signatures by extending the definition of verifiably encrypted signatures [36] to so-called one-time verifiable encrypted signatures. We refer to Section 5.3 for further details.

tion. Atomic swaps, on the other hand, allow two (or more) parties to atomically exchange tokens, i.e., either the exchange terminates and both parties obtain the other party's token or none does. Both of these applications rely on a technique that allows exchanging a secret value for a signature, which is exactly the functionality that adaptor signatures provide.

Unfortunately, despite the increasing popularity of adaptor signatures, no prior work analyzed how this primitive can be securely used in practice inside a cryptographic wallet. This is, however, of particular importance for adaptor signatures, as their security does not only rely on the secure storage of a signing secret key but also on secret witnesses.

5.1. Background on Adaptor Signatures

Before we detail our contribution of this chapter, we first recall the notion of a hard relation [65] and adaptor signatures as defined by Aumayr et al. [15]. At a high level, a relation R is said to be hard if given a statement Y it is infeasible to find a witness y such that $(Y, y) \in \mathsf{R}$. An adaptor signature scheme ASig, is defined w.r.t. a hard relation R and a signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ (cf. Definition 2.2.1) and consists four algorithms $\mathsf{ASig} = (\mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Extract})$. At a high level, for a signing public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ the algorithms pSign allows to generate an incomplete signature $\tilde{\sigma}$, a so-called pre-signature, w.r.t. a message m, a statement Y, and the secret key sk, while $\mathsf{pVerify}$ allows to verify that the pre-signature $\tilde{\sigma}$ was generated correctly for m, Y, and pk. The Adapt algorithm then allows to complete $\tilde{\sigma}$ into a full valid signature σ using a witness y where $(Y, y) \in \mathsf{R}$. Finally, the $\mathsf{Extract}$ algorithm allows to extract a witness $\tilde{y}$ such that $(Y, \tilde{y}) \in \mathsf{R}$ given the full and the pre-signature, namely σ and $\tilde{\sigma}$. An adaptor signature scheme must satisfy three security properties, namely *unforgeability*, *witness extractability*, and *pre-signature adaptability*.

At a high level, the notion of unforgeability guarantees that an adversary cannot forge a valid signature for a message m even after receiving a pre-signature for m. More concretely, unforgeability of an adaptor signature scheme is defined via a security game between a challenger and an adversary, where the adversary receives access to a signing and a pre-signing oracle which respectively return signatures and pre-signatures for messages (and statements in case of the pre-signing oracle) chosen by the adversary. Eventually, the adversary outputs a message m^* to the challenger, which in turn samples a fresh statement/witness pair (Y^*, y^*) and sends a pre-signature on m^* w.r.t. Y^* to the adversary. The adversary wins the game if (1) it outputs a valid forgery for message m^*, and if (2) the message m^* was never queried to any oracle.

Witness extractability, on the other hand, guarantees that given a pre-signature $\tilde{\sigma}$ for a message m and w.r.t. a statement Y, as well as the corresponding adapted full signature σ, the algorithm $\mathsf{Extract}$ outputs a witness $\tilde{y}$ such that $(Y, \tilde{y}) \in \mathsf{R}$. The corresponding security game proceeds as follows: similarly to the unforgeability game, the adversary receives access to a signing and pre-signing oracle and eventually outputs a message m^*. However, in contrast to the unforgeability game, the adversary outputs a statement Y^* along with m^*. The challenger then sends a pre-signature $\tilde{\sigma}$ for m^* and Y^* to the adversary, which wins the game if it can output a valid forgery σ for message m^* such that (1) the $\mathsf{Extract}$ algorithm on input $\tilde{\sigma}$ and σ outputs a witness $\tilde{y}^*$ such that $(Y^*, \tilde{y}^*) \notin \mathsf{R}$, and (2) no oracle was previously queried on m^*.

Finally, pre-signature adaptability guarantees that a pre-signature $\tilde{\sigma}$ that is valid w.r.t. a message m and statement Y (i.e., the $\mathsf{pVerify}$ algorithm on input $\tilde{\sigma}$, m, and Y outputs 1), can be adapted to a valid full signature σ for message m by executing algorithm Adapt on input $\tilde{\sigma}$ and a witness $\tilde{y}$ if $(Y, \tilde{y}) \in \mathsf{R}$.

5.2. Our Contribution

In this thesis, we initiate the study of deterministic wallets in the hot/cold setting with support for adaptor signatures. This contribution is contained in the following publication that can be found in Appendix D:

[83] A. Erwig and S. Riahi. "Deterministic Wallets for Adaptor Signatures". In: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II.* 2022, pp. 487–506. **Part of this thesis.**

In this publication, we first introduce the notion of *adaptor signatures with rerandomizable keys*, and we show an instantiation from the ECDSA signature scheme. We then discuss the design of deterministic wallets in the hot/cold setting with adaptor signature support, which we call *adaptor wallets*, and we show how to generically construct such adaptor wallets from any adaptor signature scheme with rerandomizable keys and from a witness rerandomizable hard relation The latter is a novel notion that we introduce in the above publication. Finally, we argue that it is impossible to construct an adaptor wallet from a hard relation where (1) statements and witnesses can be rerandomized independently of each other, and (2) the witness rerandomization algorithm is invertible. In the full version of the above publication [82], we additionally formalize our model for adaptor wallets, provide security arguments for our generic adaptor wallet construction, and we formalize our impossibility result. In this thesis, however, we focus only on the contribution contained in the proceedings version of our publication [83], which we detail in Section 5.2.1.

As a second contribution, we investigate how to generically construct adaptor signatures, and we extend the notion of adaptor signatures to the two-party setting within the following publication that can be found (with some adjustments[3]) in Appendix E:

[77] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. "Two-Party Adaptor Signatures from Identification Schemes". In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I.* 2021, pp. 451–480. **Part of this thesis.**

In more detail, our contribution of the above publication is as follows: as a first step, we show that we can generically construct adaptor signatures from certain signature schemes built from identification schemes via the Fiat-Shamir transformation [1, 93, 121], and we show a concrete instantiation from the Schnorr signature scheme [160]. We additionally show that adaptor signatures *cannot* be constructed from unique signature schemes [107, 133, 162]. As a second step, we show that we can generically construct two-party signature schemes with aggregatable public keys from certain signature schemes built from identification schemes via Fiat-Shamir. Finally, we introduce the notion of *two-party adaptor signatures with aggregatable public keys*, and we show a generic construction, essentially combining the two previous generic constructions. In the full version of the above publication [76], we show concrete instantiations for both of our generic constructions from the Schnorr [160], Katz-Wang [118], and Guillou-Quisquater [110] signature schemes. In this thesis, we focus on the contribution contained in the proceedings version [77], which we detail in Section 5.2.2.

5.2.1. Deterministic Wallets with Adaptor Signature Support[4]

In this section, we detail our contribution w.r.t. deterministic wallets with adaptor signature support contained in [83] (cf. Appendix D).

[3]The publication in Appendix E differs slightly from publication [77]. We refer to Appendix E for details.

[4]This subsection was largely taken verbatim from [83] with some adjustments.

Adaptor Signatures with Rerandomizable Keys

As a first contribution, we introduce the notion of adaptor signatures with rerandomizable keys, show an instantiation from ECDSA, and discuss the security of our instantiation.

Model for Adaptor Signatures with Rerandomizable Keys. Similarly to standard digital signature schemes with rerandomizable keys, we define adaptor signatures with rerandomizable keys by extending regular adaptor signatures by two key rerandomization algorithms RandPK and RandSK. That is, given an adaptor signature key pair $(\mathsf{pk}, \mathsf{sk})$ and some randomness ρ, the algorithms RandPK and RandSK respectively allow to deterministically rerandomize pk and sk using ρ to obtain a new key pair $(\mathsf{pk}', \mathsf{sk}')$ such that (1) $(\mathsf{pk}', \mathsf{sk}')$ constitutes a valid adaptor signature key pair, and (2) $(\mathsf{pk}', \mathsf{sk}')$ is identically distributed to a freshly generated key pair. Naturally, we must translate the security notions of regular adaptor signatures to the rerandomizable key setting. That is, we define security notions *unforgeability under honestly rerandomizable keys*, *witness extractability under honestly rerandomizable keys*, and *pre-signature adaptability*. The first two notions extend the respective security notions of adaptor signatures by allowing the adversary to not only obtain (pre-)signatures under the secret key sk but also under secret keys that constitute honest rerandomizations of sk. Further, in our security notions the adversary can win the game by providing a forgery either under sk or under any honestly rerandomized key. We define the notion of pre-signature adaptability for adaptor signatures with rerandomizable keys identically to the corresponding notion for standard adaptor signatures, i.e., we require that if a pre-signature is valid w.r.t. to a message m, a statement Y, and a public key pk, then the pre-signature can be adapted to a full signature using a witness $\tilde{y}$ where $(Y, \tilde{y}) \in \mathsf{R}$.

ECDSA-based Adaptor Signature with Rerandomizable Keys. As a next step, we show an instantiation of an adaptor signature scheme with rerandomizable keys by transforming the existing ECDSA-based adaptor signature scheme [15, 145] (which we denote here by AEC) into an adaptor signature with rerandomizable keys (which we denote here by RAEC). To do so, we essentially extend the AEC scheme by algorithms RandPK and RandSK which multiplicatively rerandomize the scheme's key pair. More concretely, the algorithms RandPK and RandSK rerandomize a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ with a randomness ρ by multiplying ρ to the key pair, i.e., $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$ and $\mathsf{sk}' \leftarrow \mathsf{sk} \cdot \rho$. We use multiplicative rerandomization rather than an additive one for the following reason: As we discussed in Chapter 3, the standard ECDSA signature scheme with additive key rerandomization incurs a security loss in the number of rerandomized keys, whereas the ECDSA signature scheme with multiplicative rerandomization of Das et al. [68] does not incur such a loss. This security loss stems from the related key attack that is required to prove the security of the additively rerandomizable scheme. Since the security proof for the ECDSA-based adaptor signature with rerandomizable keys would rely on the same related key attack, a similar security loss can be expected. Worse yet, the related key attack for additively rerandomizable ECDSA allows to prove only one-per-message unforgeability (cf. Chapter 3). Therefore, we use multiplicative rerandomization in our instantiation.

Security Analysis of Our Construction. To analyze the security of our RAEC scheme, we follow the approach of Das et al. [68], who presented a security proof of the plain ECDSA signature scheme with multiplicatively rerandomizable keys via a reduction to the (non-rerandomizable) ECDSA signature scheme. The main ingredient in their security proof is an RKA which allows to transform a signature on message m under a public key pk to a signature for a message m' under the related public key $\mathsf{pk}' \leftarrow \rho \cdot \mathsf{pk}$ if the randomness ρ has a certain structure (cf. Chapter 3). This RKA allows their reduction to answer signing queries under rerandomized keys and to transform a forgery under a rerandomized public key pk' to a forgery under the original public key pk.

In our case, we provide proof sketches for a reduction from the unforgeability of our RAEC scheme to the unforgeability of the AEC scheme and a reduction from the witness extractability of our RAEC scheme to the witness extractability of the AEC scheme. However, these reductions do not only require an RKA to transfrom full ECDSA signatures (as was done by Das et al. [68]), but also an RKA to transform pre-signatures. We therefore show that the RKA of Das et al. can also be applied to our RAEC scheme to transform pre-signatures. With this RKA for full- and pre-signatures in place, we show proof sketches for the two reductions. The property of pre-signature adaptability of our RAEC scheme follows directly from the corresponding property of the AEC scheme.

Adaptor Wallets

We next introduce the idea of *adaptor wallets*, which are essentially deterministic wallets in the hot/cold setting with adaptor signature support. For our notion of adaptor wallets we broadly follow the BIP32 standard of hierarchical deterministic wallets (cf. Chapter 3) in the sense that we consider a master wallet which can deterministically derive either hardened or non-hardened child wallets, and we assume that non-hardened wallets are implemented in the hot/cold setting.

However, our adaptor wallet model crucially differs from the notion of hierarchical deterministic wallets according to BIP32 in the following three points: (1) since adaptor wallets must be able to support adaptor signature functionality, they must not only store signing key pairs, but also statement/witness pairs; (2) similarly to the signing key pairs, the statement/witness pairs must be deterministically derived from the parent to the child wallet, i.e., we require a deterministic derivation mechanism for statement/witness pairs; (3) for simplicity, we do not allow child wallets to initialize further child wallets as is done in the fully hierarchical setting. For non-hardened wallets that we assume to be implemented in the hot/cold setting, we assume that the cold wallet stores the node's witness and the hot wallet stores the corresponding statement.

In more detail, the design of our adaptor wallets is as follows: the master wallet initially generates and stores a master key pair $(\mathsf{pk}, \mathsf{sk})$ of an adaptor signature scheme with rerandomizable keys, a state St, and a master statement/witness pair (Y_m, y_m) of a hard relation. It can then deterministically initialize a child wallet with identifier ID by deriving a new key pair $(\mathsf{pk}^{\mathsf{ID}}, \mathsf{sk}^{\mathsf{ID}})$ from $(\mathsf{pk}, \mathsf{sk})$ and St, as well as a new statement/witness pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ from (Y_m, y_m). The child wallet can then initialize a counter ctr and use it together with the pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ to deterministically derive further statement/witness pairs $(Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}})$. The derivation of these pairs is necessary since adaptor signature applications (such as payment channels or atomic swaps) typically require the use of several statement/witness pairs, i.e., the pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ might not be sufficient for the execution of all adaptor signature applications. We illustrate our adaptor wallet design in Figure 5.1.

Derivation of Statement/Witness Pairs. Clearly, for an adaptor wallet scheme as described above we require a mechanism which allows to deterministically derive statement/witness pairs. Ideally, this mechanism would allow to derive child statements and witnesses independently, such that a child statement, say $Y^{\mathsf{ID}}_{\mathsf{ctr}}$, could be derived only from the parent statement Y^{ID}, the counter ctr and some state. Analogously a child witness, say $y^{\mathsf{ID}}_{\mathsf{ctr}}$, could be derived only from the parent witness y^{ID}, ctr, and the same state. Such a mechanism would integrate perfectly into the hot/cold setting, where the cold wallet stores only the witness and the hot wallet stores only the statement and both wallets can independently derive further child witnesses or statements without having to interact with each other. Surprisingly, we show that for the discrete logarithm hard relation[5], such an independent

[5]In our work, we are mostly interested in the discrete logarithm hard relation $\mathsf{R}^{dlog} := \{(Y, y) \mid Y = g^y\}$ because the adaptor signature variants of the two signature schemes that are most widely used in cryptocurrency networks, namely ECDSA and Schnorr, rely on this (or a similar) relation.

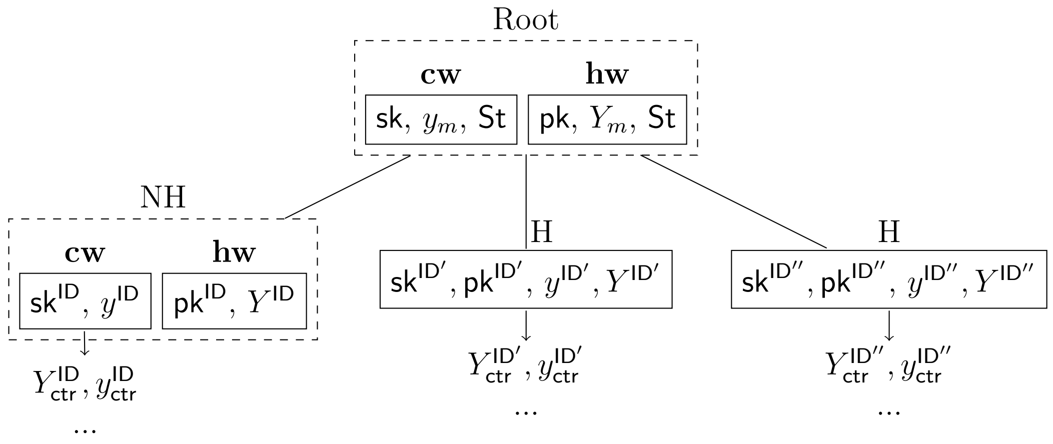

Root
cw
hw
sk, y_m, St
pk, Y_m, St
NH
H
H
cw
hw
$\mathsf{sk}^{\mathsf{ID}}$, y^{ID}
$\mathsf{pk}^{\mathsf{ID}}$, Y^{ID}
$\mathsf{sk}^{\mathsf{ID}'}, \mathsf{pk}^{\mathsf{ID}'}, y^{\mathsf{ID}'}, Y^{\mathsf{ID}'}$
$\mathsf{sk}^{\mathsf{ID}''}, \mathsf{pk}^{\mathsf{ID}''}, y^{\mathsf{ID}''}, Y^{\mathsf{ID}''}$
$Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}}$
$Y^{\mathsf{ID}'}_{\mathsf{ctr}}, y^{\mathsf{ID}'}_{\mathsf{ctr}}$
$Y^{\mathsf{ID}''}_{\mathsf{ctr}}, y^{\mathsf{ID}''}_{\mathsf{ctr}}$
...

Figure 5.1.: Design of our adaptor wallet scheme with three child wallets. H and NH denote hardened and non-hardened nodes respectively, **cw** and **hw** denote cold and hot wallets respectively and the values below the child wallets (e.g. $y^{\mathsf{ID}}_{\mathsf{ctr}}, Y^{\mathsf{ID}}_{\mathsf{ctr}}$) illustrate the statement/witness pairs that are being derived within each child wallet using an internal counter ctr. Figure taken from [83].

derivation is impossible if statements/witnesses are derived multiplicatively or additively. At a high level, that is because adaptor signature applications typically require child witnesses to be leaked at some point during the execution of the application. Therefore, if the derivation mechanism is invertible (as is the case for additive and multiplicative derivation), an adversary could compute the parent witness from the child witness. As an example, consider a simple multiplicative derivation mechanism, where the child witness $y^{\mathsf{ID}}_{\mathsf{ctr}}$ is computed from the parent witness y^{ID} as $y^{\mathsf{ID}}_{\mathsf{ctr}} \leftarrow y^{\mathsf{ID}} \cdot \rho$. Then an adversary knowing $y^{\mathsf{ID}}_{\mathsf{ctr}}$ and ρ could trivially compute y^{ID} and thereby break the security of the adaptor wallet.

In our model and construction, we therefore resort to a derivation mechanism, where the parent witness y^{ID} is required even for the derivation of a child statement $Y^{\mathsf{ID}}_{\mathsf{ctr}}$. To this end, we define the notion of a *witness rerandomizable* hard relation which allows to deterministically derive a child statement/witness pair (Y', y') from a parent witness y and some randomness ρ. More concretely, a witness rerandomizable hard relation extends the notion of a hard relation by a deterministic algorithm $\mathsf{RandWit}$ that on input a witness y and some randomness ρ outputs a new witness y', from which the corresponding statement Y' can be computed. A witness rerandomizable hard relation must satisfy perfect rerandomizability, which guarantees that if ρ was chosen uniformly at random from an appropriate randomness space, then the derived statement/witness pair (Y', y') is identically distributed to a freshly generated statement/witness pair.

The consequence of using such a witness rerandomizabe hard relation is that in our adaptor wallet model, the cold wallet of a non-hardened node, which stores the node's witness, is required to come online even for the derivation of a child statement. While this is an important limitation, we argue that our scheme is still sufficient to execute the two main applications of adaptor signatures, namely atomic swaps and payment channels. For instance, the application of atomic swaps typically requires the involved parties to generate only few signatures and/or statement/witness pairs. Therefore, a cold wallet that is used for the execution of an atomic swap does not need to be activated frequently. Further, in practice one can minimize the number of times a cold wallet must be activated by batching the generation of statement/witness pairs, i.e., the cold wallet can generate multiple pairs and send all statements at once to the hot wallet. For other applications with frequent transactions, such as payment channels, an adaptor wallet user can simply use a hardened node.

5.2.2. Two-Party Adaptor Signatures

In this section, we detail the contribution of our publication [77] (cf. Appendix E). The core contribution of this publication is the formulation of two-party adaptor signatures which extend standard adaptor signatures to the setting, where two parties can jointly generate (pre-)signatures. In more detail, we introduce the notion of two-party adaptor signatures with aggregatable public keys which allows two parties, say Alice (holding a key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$) and Bob (holding a key pair $(\mathsf{pk}_B, \mathsf{sk}_B)$), to jointly generate (pre-)signatures under an aggregated public key pk_{AB} that can be computed from pk_A and pk_B.

A motivating example for such a primitive are payment channels. At a high level, a payment channel allows two parties to send many payments to each other without having to involve the blockchain and therefore without incurring fees or confirmation delays. In a bit more detail, Alice and Bob can open a payment channel by locking some $c = c_A + c_B$ coins on the blockchain where Alice initially owns c_A coins and Bob owns c_B coins. That is, the initial state of the channel is (c_A, c_B). Once the channel is open, Alice and Bob can issue payments to each other without having to involve the blockchain by essentially updating the state of the channel. More concretely, if Alice sends $t \leq c_A$ coins to Bob, then both parties update the state of the channel to $(c_A - t, c_B + t)$ by exchanging signatures on this new coin distribution. In order to close the channel, one of the parties sends the latest state to the blockchain which unlocks the c coins and distributes them accordingly. Importantly, each channel state must always be signed by each party, i.e., when the channel is closed two signatures must be sent to the blockchain which increases the transaction size and consequently the fees both parties have to pay during channel closure. The two signatures are required to prove that *both* parties agreed to the channel state. However, Alice and Bob could use a two-party signature scheme instead to jointly generate a *single* signature for each state, which then decreases the amount of fees during channel closure. Recently, some works have considered to use adaptor signatures instead of standard signature schemes for the construction of channels (e.g., [15, 167]). To achieve the same fee benefits as described above for these channels, we require a two-party adaptor signature scheme. We now provide an overview of our contribution in more detail.

Generic Construction of Adaptor Signatures

As a first contribution, we explore the possibility of generically constructing adaptor signatures. To this end, we show that signature schemes that are constructed from identification (ID) schemes via the Fiat-Shamir transformation can be used to construct adaptor signatures if the signature schemes satisfy certain conditions. An identification scheme is a well-known primitive in cryptography which allows a party to prove knowledge of a secret key to a verifier in an interactive protocol. In our work, we focus on so-called *canonical* identification schemes [1, 121] which are three-move protocols: First, the prover sends a commitment R to the verifier, which in turn replies with a challenge h. In the last move, the prover sends a response s. Using the transcript (R, h, s) and the prover's public key, the verifier can finally check whether the prover indeed knows the correct secret key. Such canonical ID schemes can be transformed into standard digital signature schemes via the Fiat-Shamir transformation [1, 93, 121]. In our work, we show that we can further transform such signature schemes (which we denote here by IDSig) into adaptor signature schemes if the IDSig schemes satisfy certain conditions. More concretely, we require an IDSig scheme to satisfy the following three conditions:

1. There must exist a deterministic function $\mathsf{f}_{\mathsf{shift}}$, which takes as input a commitment R and a statement Y of a hard relation R, and outputs a new commitment R'.

2. There must exist a deterministic function $\mathsf{f}_{\mathsf{adapt}}$, which takes as input a response s for a commitment/challenge pair (R, h) and a witness y of a hard relation R. It outputs a new response s' for a commitment/challenge pair (R', h) where $R' \leftarrow \mathsf{f}_{\mathsf{shift}}(R, Y)$ and $(Y, y) \in \mathsf{R}$.

3. There must exist a deterministic function $\mathsf{f}_{\mathsf{ext}}$, which takes as input a response s for a commitment/challenge pair (R, h) and a response s' for a commit-

ment/challenge pair (R', h) where $R' \leftarrow \mathsf{f_{shift}}(R, Y)$. The function outputs a witness y' such that $(Y, y') \in \mathsf{R}$.

At a high level, the above conditions are necessary to ensure that the IDSig scheme can be extended to support the additional adaptor signature functionality. We then show a generic construction of adaptor signature schemes from any IDSig scheme that satisfies the above conditions, and we prove our generic construction secure in the random oracle model. That is, we prove that our generic construction satisfies unforgeability, witness extractability, and pre-signature adaptability. Finally, we show that our generic construction can be instantiated from Schnorr signatures [160]. As an additional result, we show that it is impossible to construct an adaptor signature scheme from unique signatures [107, 133, 162].

Generic Construction of Two-Party Signatures from ID Schemes

As a second contribution, we show that we can generically transform an IDSig scheme to a *two-party signature scheme with aggregatable public keys* (which we denote here by SigTwo) if the IDSig scheme satisfies certain conditions. At a high level, a two-party signatue scheme with aggregatable public keys allows two parties P_0 and P_1, which each hold a public/secret key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively, to jointly compute a signature for some message m via an interactive signing protocol, such that the resulting signature is valid w.r.t. a combined public key pk that can be computed from pk_0 and pk_1. In a bit more detail, a two-party signature scheme with aggregatable public keys consists of (1) a setup algorithm, which sets up the initial public parameters, (2) a key generation algorithm, (3) an interactive signing procedure, (4) a public key aggregation algorithm, which allows to aggregate two public keys into one aggregated public key, and (5) a verification algorithm. For our definition of SigTwo schemes, we closely follow the definition of multi-signatures with aggregatable public keys which have previously been introduced and used in several works (e.g., [35, 126, 141]). A SigTwo scheme must satisfy two properties, namely completeness and unforgeability. The first property is a correctness property, which says that if parties P_0 and P_1 honestly execute the signing procedure for a message m and w.r.t. their respective key pair, then the resulting signature verifies w.r.t. m and the combined public key as output by the key aggregation algorithm on input pk_0 and pk_1. The second property guarantees that even if one of P_0 or P_1 is malicious and can execute the interactive signing procedure jointly with the respective other party on adaptively chosen messages, the malicious party cannot forge a valid signature under the combined public key for a fresh message. We define the corresponding unforgeability game in the so-called knowledge of secret key model (KOSK) [33] where the adversary may sample its own key pair at the beginning of the game, but must send its public and secret key to the challenger. We use this model in order to avoid the use of rewinding in our security proofs.

We show how to generically construct a SigTwo scheme from any IDSig scheme that satisfies certain conditions. We denote our generic construction of a SigTwo scheme in the following by $\mathsf{IDSigTwo}$. At a high level, the joint signing procedure of $\mathsf{IDSigTwo}$ proceeds as follows: Recall that the signing procedure is executed between two parties P_0 and P_1, which each obtain as input a public/secret key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively. During the signing procedure, each party initially computes a commitment value R_0 and R_1 respectively, which they exchange and combine into a single commitment R. Using R, each party computes the corresponding challenge h, and finally each party computes a signature share s_0 or s_1 respectively. Ultimately, the two parties exchange their signature shares and combine them into one full signature s, which constitutes a valid response for the combined public key as output by the public key aggregation algorithm on input pk_0 and pk_1. With this high level overview in mind, we now explain the three conditions that an IDSig scheme must satisfy for our generic construction of $\mathsf{IDSigTwo}$:

1. There must exist a deterministic function $\mathsf{f}_{\mathsf{comb-rand}}$, which takes as input two commitments R_0 and R_1, and outputs a combined commitment R.

2. There must exist a deterministic function $\mathsf{f}_{\mathsf{comb-sig}}$, which takes as input two responses s_0 and s_1 for commitment/challenge pairs (R_0, h) and (R_1, h) respectively. The function outputs a combined response s for the commitment/challenge pair (R, h) where $R \leftarrow \mathsf{f}_{\mathsf{comb-rand}}(R_0, R_1)$.

3. There must exist a deterministic function $\mathsf{f}_{\mathsf{comb-pk}}$, which takes as input two public keys pk_0 and pk_1, and outputs a combined public key pk. It must hold that for two commitment/challenge/response tuples (R_0, h, s_0) and (R_1, h, s_1) which are respectively valid under pk_0 and pk_1, the tuple (R, h, s) where $s \leftarrow \mathsf{f}_{\mathsf{comb-sig}}(s_0, s_1)$ and $R \leftarrow \mathsf{f}_{\mathsf{comb-rand}}(R_0, R_1)$ is valid under $\mathsf{pk} \leftarrow \mathsf{f}_{\mathsf{comb-pk}}(\mathsf{pk}_0, \mathsf{pk}_1)$.

4. There exists a deterministic function $\mathsf{f}_{\mathsf{decomb-sig}}$, which takes as input a challenge/response pair (h, s) and a secret key sk, and outputs a challenge/response pair (h, s'). It must hold that if (R, h, s) is valid under $\mathsf{pk} \leftarrow \mathsf{f}_{\mathsf{comb-pk}}(\mathsf{pk}_0, \mathsf{pk}_1)$, then $(R, h, s_b) \leftarrow \mathsf{f}_{\mathsf{decomb-sig}}((h, s), \mathsf{sk}_{1-b})$ is valid under pk_b for $b \in \{0, 1\}$.

Finally, we prove in the random oracle model that our generic $\mathsf{IDSigTwo}$ construction satisfies unforgeability via reduction from the unforgeability property of the underlying IDSig scheme. We essentially show that if an efficient adversary can come up with a valid forgery in IDSigTwo, then we can construct an efficient adversary that can use this forgery and the function $\mathsf{f}_{\mathsf{decomb-sig}}$ to produce a valid forgery for the IDSig scheme. In our reduction, we simulate the signing procedure of the IDSigTwo scheme by appropriately programming the random oracle.

Two-Party Adaptor Signatures with Aggregatable Public Keys

As a final contribution, we introduce the primitive of *two-party adaptor signatures with aggregatable public keys*, which essentially combines the definitions of adaptor signatures and two-party signatures with aggregatable public keys. That is, we define a two-party adaptor signature scheme with aggregatable public keys (which we denote here by ASigTwo) with respect to a SigTwo scheme and a hard relation R. An ASigTwo scheme then additionally defines (1) an interactive pre-signing protocol, which can be jointly executed by parties P_0 and P_1 to generate a pre-signature for a message m and a statement Y, and (2) algorithms pVerify, Adapt, and Extract which are defined in the same way as for standard adaptor signature schemes (cf. Section 5.1).

We define the properties of *two-party pre-signature correctness*, *unforgeability*, *two-party witness extractability*, and *two-party pre-signature adaptability* for an ASigTwo scheme. At a high level, these properties guarantee the following:

- *Two-Party Pre-Signature Correctness:* If parties P_0 and P_1 honestly execute the pre-signing protocol on input their respective key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$, a message m, and a statement Y, then the resulting pre-signature $\tilde{\sigma}$ can be adapted to a valid full signature σ under the combined public key using witness y with $(Y, y) \in \mathsf{R}$. In addition, the pre-signature $\tilde{\sigma}$ must pre-verify (i.e., the pVerify algorithm must output 1), and executing the Extract algorithm on input σ, $\tilde{\sigma}$, and Y must output a witness y' such that $(Y, y') \in \mathsf{R}$.

- *Unforgeability and Two-Party Witness Extractability:* The unforgeability and two-party witness extractability notions are defined identically to the respective notions of regular adaptor signatures with the following differences: At the beginning of the respective security games, the adversary is allowed to corrupt either P_0 or P_1 and to control that party throughout the entire game. In addition, the (pre-)signing oracles are now interactive, i.e., the respective oracles execute the (pre-)signing protocols jointly with the adversary, where the oracles execute the role of the honest party.

- *Two-Party Pre-Signature Adaptability:* The two-party pre-signature adaptability property guarantees that a pre-signature $\tilde{\sigma}$ that is valid for a message

m, a statement Y, and under a combined public key pk, can be adapted to a full signature σ for message m and under pk by executing the algorithm Adapt on input $\tilde{\sigma}$ and a witness y' where $(Y, y') \in \mathsf{R}$.

We then show how to generically construct an $\mathsf{ASigTwo}$ scheme from our generic $\mathsf{IDSigTwo}$ construction as presented above. The security of our construction can be shown by combining our security proofs for our generic adaptor signature construction and for our generic two-party signature with aggregatable public keys construction.

5.3. Related Work

We now provide an overview over relevant related works for the topics of adaptor signatures as well as multi-signatures and identification schemes. For a comprehensive overview over related works for cryptographic wallets, we refer to Section 3.2.1.

5.3.1. Adaptor Signatures

Poelstra [152] first introduced the primitive of adaptor signatures in 2017, which has found wide applications in blockchain networks since then, in particular for payment channel (hubs) (e.g., [15, 154, 167]) and for atomic swaps (e.g., [70, 112, 171]). Aumayr et al. [15] and Fournier [96] concurrently provided a formal model for adaptor signatures for the first time and proved the security of a Schnorr-based [152] instantiation in their respective models. Aumayr et al. additionally proved the security of an ECDSA-based [145] instantiation. We note that the security model of Fournier is weaker than the one of Aumayr et al. since its unforgeability notion does not allow the adversary to learn a pre-signature on the forgery message. Indeed, due to this reason, the model of Fournier is not suitable for certain blockchain applications.

Dai et al. [63] revised the formal definition of adaptor signatures as provided by Aumayr et al. [15]. Concretely, Dai et al. introduced the security notion of (strong) full extractability, and showed that this notion strictly implies the security notions of unforgeability and witness extractability (cf. Section 5.1) for adaptor signatures. In addition, Dai et al. introduced the notion of unlinkability, which guarantees at a high level that an adapted pre-signature for a message m is indistinguishable from a signature computed via the signing algorithm for the same message m. The authors also showed a generic construction of adaptor signatures from any signature scheme and any hard relation, however, with the drawback that the resulting adaptor signature scheme produces signatures with a different structure than those of the underlying signature scheme. This is a significant caveat, since it essentially renders the resulting adaptor signatures incompatible with any major blockchain network. Recently, Tu et al. [176] presented two slightly modified variants of the previous ECDSA-based adaptor signature scheme [15, 145] and argued that these variants can be used for efficient atomic swaps in certain scenarios.

Another line of work focused on adaptor signature schemes with post-quantum security [84, 104, 122, 168]. More concretely, Esgin et al. [84] introduced an adaptor signature scheme based on lattice assumptions, namely Module-SIS and Module-LWE [39, 125], whereas Tairi et al. [168] introduced an adaptor signature scheme based on an isogeny assumption, namely the MT-GAIP assumption [90]. Similarly, within a Master's thesis, Gilchrist [104] constructed an adaptor signature scheme from the SQISign signature scheme [91] whose security is based on isogeny assumptions. Finally, Klamti and Hasan [122] proposed an adaptor signature scheme based on assumptions from coding theory.

Several works considered schemes with a similar functionality as adaptor signatures (e.g., [7, 19, 57, 170]). In particular, Thyagarajan and Malavolta [170] introduced the notion of lockable signatures, which allows to generate a *lock* from two signatures σ and $\tilde{\sigma}$ such that (1) the lock hides σ, and (2) signature σ can be recovered from the lock knowing $\tilde{\sigma}$. While conceptually similar, lockable signatures are a weaker primitive than adaptor signatures as the generation of a lock requires knowledge of the unlocking secret, namely signature $\tilde{\sigma}$. For adaptor signatures, however, the generation of a pre-signature (which is the equivalent of a lock) requires only knowledge of a publicly known statement, but not of a corresponding secret witness. Chen et al. [57] introduced the notion of concurrent signatures, which allows two parties to exchange incomplete signatures that require an additional value to be completed to full signatures. However, concurrent signatures only support the exchange of two signatures, whereas adaptor signatures allow to exchange a signature for an arbitrary witness of the respective hard relation. Furthermore, concurrent signatures require a change in the signature verification algorithm of the underlying signature scheme which makes the primitive unsuitable for the blockchain setting.

Finally, various works (e.g., [154, 166]) extended the notion of adaptor signatures to more complex primitives which were then used for the construction of blockchain applications.

5.3.2. Multi-Signatures and ID Schemes

In the past, there has been extensive work on the topic of multi-signatures (e.g., [22, 115, 142, 151]), which allow a set of signers, who each hold a signing public/secret key pair, to collaboratively generate a signature on some message m such that the resulting signature verifies w.r.t. m and the public keys of all signers. Several works also considered the extension of multi-signatures with aggregatable public keys (e.g. [35, 126, 141]), where the public keys of the signers can be aggregated into a single joint public key. The latter is particularly interesting for the blockchain setting since (1) the aggregated public key is short and hence can be easily stored on the blockchain, and (2) the resulting signature can be verified via the verification algorithm of the underlying (single party) signature scheme that the multi-signature scheme builds upon.

There exists a large body of works on canonical identification schemes (ID schemes) such as [27, 93, 110, 143], and several works studied the security of signature schemes built from ID schemes via the Fiat-Shamir transform [93] (e.g., [1, 23, 119, 121]). Our results on generic constructions of adaptor signatures, two-party signatures with aggregatable public keys, and two-party adaptor signatures with aggregatable public keys from ID schemes as contained in our publication [77] partly build upon the work of Kiltz et al. [121], which gave a concrete security analysis of such signature schemes obtained from ID schemes.

6. Conclusion

Cryptographic wallets are an essential tool for cryptocurrency users to securely store and maintain their secret signing keys and thereby to protect their funds from attackers. This thesis contributes generally to the advancement of provably secure cryptographic wallets and their use in cryptocurrency and blockchain networks. In particular, in this thesis we analyzed and improved upon the BIP32 standard [179], the current state of the art standard for hierarchical deterministic wallets. We will conclude this thesis by providing a discussion on interesting open research questions in the context of cryptographic wallets.

Custodial-Specific Wallets. The custody type of a cryptographic wallet determines who is in control of the wallet and thereby of the funds that the wallet stores. All wallet schemes that we presented in this thesis are agnostic of whether they are being used as custodial or non-custodial wallet, i.e., the wallets can either be maintained by a service provider on behalf of a user or by the user itself. While this custody-agnostic approach allows to use our wallet schemes in any of the two aforementioned custody settings, it might be interesting to develop wallets for a specific custody type. The advantage of this custody-specific approach is that it allows to better address the specific requirements for each setting separately. For instance, in addition to the properties of wallet unforgeability and wallet unlinkability, it might be desirable that a custodial wallet satisfies a message-privacy property. This property should roughly guarantee that the service provider, which maintains the wallet on behalf of a user, does not learn any information about the transactions that the user instructs the service provider to sign. This would be an essential property for custodial wallets to prevent censorship attacks, where the service provider refuses to sign certain transactions based on their contents (e.g., transactions to a specific address). Such a message-privacy property, however, is not required in the non-custodial setting, where the user signs all transactions itself. On the other hand, in the non-custodial setting a key backup mechanism is crucial, which allows the user to restore its signing secret key in case it loses its wallet. Hence, constructing wallet schemes for specific custodial types allows to define required security properties in a more fine-grained way and therefore to achieve more secure and potentially more efficient solutions.

As mentioned in the introduction of this thesis, a compelling middle ground between custodial and non-custodial wallets are so-called shared-custodial wallets, where a user and a service provider jointly maintain a wallet such that none can generate signatures without the help of the other. Such wallets have significant advantages over (non-)custodial wallets in the sense that (1) the user does not have to trust the service provider, and (2) the user does not lose all of its coins if its device gets corrupted. Indeed, due to these advantages, several companies such as Sepior[1] and ZenGo[2] already develop and offer a shared-custodial wallet service. We regard it as an interesting future work to provide a formal model for such wallets and to show constructions that can be proven secure in the respective model.

Deterministic Wallets for Advanced Signature Schemes. Most existing works on cryptographic wallets have focused on either standard signature schemes such as ECDSA [116], Schnorr [160], or BLS [37] (e.g., [67, 68, 111]), on their threshold variants (e.g. [30, 66, 123]), or on signature schemes that are incompatible with many blockchain networks (e.g., [114, 132, 181]). However, there exist various more advanced notions of signature schemes which extend the functionality of standard digital signature schemes and which are often found to be useful in the context of blockchain networks. One example for such an advanced signature scheme are adaptor signatures [15, 96, 152] for which we introduced a deterministic wallet in this thesis. Unfortunately, several other advanced signature schemes have not yet been analyzed in the context of cryptographic wallets. For instance, in our publication [77], we introduced the notion of two-party adaptor signatures with aggregatable public keys which allow for the construction of more cost-efficient channels. Similarly, Qin et al. [154] introduced the notion of blind adaptor signatures and show that this primitive can be used to construct Bitcoin-compatible anonymous payment channel hubs. Verifiable timed signatures [169] are a recently proposed primitive that find applications in blockchain networks for privacy-preserving payment channel networks (e.g., [135, 136]), multi-signature transactions, and fair multiparty computation with financial compensation. Unfortunately, for none of the above primitives it has been analyzed how they can be securely executed within a cryptographic wallet. We regard this as an interesting future work, since these schemes will ultimately only be useful in practice, if they can be executed in a secure environment and if users can securely store the involved secret values.

[1] `https://sepior.com/`
[2] `https://zengo.com/`

Two-Factor Authenticated Password Wallets. Two-factor authentication is a widely used mechanism to significantly increase the security of a system in practice. The idea behind this mechanism is that, in order to break the security of a system, an adversary must compromise two different parts of the system. As an example, consider a traditional banking system: in order to obtain access to a user's funds, an adversary must (1) compromise the banking device of the user (e.g., its smartphone or banking card), and (2) know the user's password or PIN. Only one of these two factors is not sufficient to compromise the user's banking account.

A similar mechanism would be useful to strengthen the security of cryptographic wallets. In some sense, existing threshold wallets such as [66, 123] inherently offer a multi-factor authentication mechanism, since a certain threshold of wallet devices is necessary to generate a valid signature and therefore to spend funds. However, securely maintaining several devices and activating a subset of them each time a user wants to send a payment is a cumbersome task in practice. A more common way to implement a two-factor authentication mechanism (which is indeed also used by traditional banks as described above) is to combine the access to a device with the knowledge of a password. Indeed, already in 2012, the Bitcoin community acknowledged the need for password-protected secret keys within the BIP38 standard [44]. However, even though various password-based cryptographic primitives have been thoroughly studied in the past (e.g., [17, 25, 45]), there exist only few works in the context of wallets. To the best of our knowledge, the only works that propose the use of a password or passphrase for wallets consider only the very specific case of hardware wallets [11, 139].

We believe that it is an interesting research question to develop an authentication mechanism based on passwords that can be generically combined with existing wallet schemes. Essentially, such a generic construction would yield a framework that allows to readily strengthen the security of a cryptographic wallet scheme by adding a two-factor authentication mechanism. Such a mechanism would be particularly useful for our deterministic wallet constructions that we presented in this thesis: recall that in our constructions we often assumed cold wallets to be incorruptible since the leakage of a secret key can trivially break the security of the entire wallet scheme. Now assume we have a mechanism that allows to store the secret key in the cold wallet encrypted under a password. Then, to break the security of the wallet scheme, an adversary must not only corrupt the cold wallet, but also know the correct password under which the secret key is encrypted. That is, using such a password-based authentication mechanism, we might be able to prove our wallet constructions secure in a stronger model, where we allow an adversary to even corrupt cold wallets.

7. Bibliography

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. “From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security”. In: *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings.* 2002, pp. 418–433.

[2] *Abelian.* `https://www.abelian.info/`.

[3] D. Abram, A. Nof, C. Orlandi, P. Scholl, and O. Shlomovits. “Low-Bandwidth Threshold ECDSA via Pseudorandom Correlation Generators”. In: *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022.* 2022, pp. 2554–2572.

[4] D. Aggarwal, G. Brennen, T. Lee, M. Santha, and M. Tomamichel. “Quantum Attacks on Bitcoin, and How to Protect Against Them”. In: *Ledger* (2018).

[5] N. A. Alkadri, P. Das, A. Erwig, S. Faust, J. Krämer, S. Riahi, and P. Struck. “Deterministic Wallets in a Quantum World”. In: *CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.* 2020, pp. 1017–1031. **Part of this thesis.**

[6] E. Alkim, P. S. L. M. Barreto, N. Bindel, J. Krämer, P. Longa, and J. E. Ricardini. “The Lattice-Based Digital Signature Scheme qTESLA”. In: *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I.* 2020, pp. 441–460.

[7] O. Alpos, Z. Wang, A. Kavousi, S. Y. Chau, D. Le, and C. Cachin. “DSKE: Digital Signature with Key Extraction”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 1753.

[8] J. Alupotha and X. Boyen. “Practical UC-Secure Zero-Knowledge Smart Contracts”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 670.

[9] J. Alupotha, X. Boyen, and M. McKague. “Aggregable Confidential Transactions for Efficient Quantum-Safe Cryptocurrencies”. In: *IEEE Access* (2022), pp. 17722–17747.

[10] A. Ambainis, M. Hamburg, and D. Unruh. “Quantum Security Proofs Using Semi-classical Oracles”. In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II.* 2019, pp. 269–295.

[11] M. Arapinis, A. Gkaniatsou, D. Karakostas, and A. Kiayias. “A Formal Treatment of Hardware Wallets”. In: *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers.* 2019, pp. 426–445.

[12] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken. “Dynamic and Efficient Key Management for Access Hierarchies”. In: *ACM Trans. Inf. Syst. Secur.* 3 (2009), 18:1–18:43.

[13] N. Atzei, M. Bartoletti, and T. Cimoli. “A Survey of Attacks on Ethereum Smart Contracts (SoK)”. In: *Principles of Security and Trust - 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings.* 2017, pp. 164–186.

[14] J. Aumasson, A. Hamelink, and O. Shlomovits. “A Survey of ECDSA Threshold Signing”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 1390.

[15] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi. “Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures”. In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II.* 2021, pp. 635–664.

[16] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno-Sanchez. “Bitcoin-Compatible Virtual Channels”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* 2021, pp. 901–918.

[17] A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. “Password-protected secret sharing”. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011.* 2011, pp. 433–444.

[18] T. Bamert, C. Decker, R. Wattenhofer, and S. Welten. “BlueWallet: The Secure Bitcoin Wallet”. In: *Security and Trust Management - 10th International Workshop, STM 2014, Wroclaw, Poland, September 10-11, 2014. Proceedings.* 2014, pp. 65–80.

[19] W. Banasik, S. Dziembowski, and D. Malinowski. "Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts". In: *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II.* 2016, pp. 261–280.

[20] J. Baron, K. E. Defrawy, J. Lampkins, and R. Ostrovsky. "Communication-Optimal Proactive Secret Sharing for Dynamic Groups". In: *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers.* 2015, pp. 23–41.

[21] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme". In: *J. Cryptol.* 3 (2003), pp. 185–215.

[22] M. Bellare and G. Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006.* 2006, pp. 390–399.

[23] M. Bellare, B. Poettering, and D. Stebila. "From Identification to Signatures, Tightly: A Framework and Generic Transforms". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II.* 2016, pp. 435–464.

[24] M. Bellare and P. Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.* 1993, pp. 62–73.

[25] S. M. Bellovin and M. Merritt. "Encrypted key exchange: password-based protocols secure against dictionary attacks". In: *1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 4-6, 1992.* 1992, pp. 72–84.

[26] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014.* 2014, pp. 459–474.

[27] T. Beth. "Efficient Zero-Knowledge Identification Scheme for Smart Cards". In: *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings.* 1988, pp. 77–84.

[28] *Bitcoin Post-Quantum.* `https://bitcoinpq.org/`.

[29] *Bitcoin Wiki - Elliptic Curve Digital Signature Algorithm.* `https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm`.

[30] C. Blokh, N. Makriyannis, and U. Peled. "Efficient Asymmetric Threshold ECDSA for MPC-based Cold Storage". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1296.

[31] M. Blum. "Coin flipping by telephone a protocol for solving impossible problems". In: *SIGACT News* 1 (1983), pp. 23–27.

[32] M. Blum, P. Feldman, and S. Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA.* 1988, pp. 103–112.

[33] A. Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings.* 2003, pp. 31–46.

[34] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. "Random Oracles in a Quantum World". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings.* 2011, pp. 41–69.

[35] D. Boneh, M. Drijvers, and G. Neven. "Compact Multi-signatures for Smaller Blockchains". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II.* 2018, pp. 435–464.

[36] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps". In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings.* 2003, pp. 416–432.

[37] D. Boneh, B. Lynn, and H. Shacham. "Short Signatures from the Weil Pairing". In: *J. Cryptol.* 4 (2004), pp. 297–319.

[38] C. Bonte, N. P. Smart, and T. Tanguy. "Thresholdizing HashEdDSA: MPC to the Rescue". In: *Int. J. Inf. Sec.* 6 (2021), pp. 879–894.

[39] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 3 (2014), 13:1–13:36.

[40] S. Brands. “Untraceable Off-line Cash in Wallets with Observers (Extended Abstract)”. In: *Advances in Cryptology - CRYPTO ’93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*. 1993, pp. 302–318.

[41] J. Breitner and N. Heninger. “Biased Nonce Sense: Lattice Attacks Against Weak ECDSA Signatures in Cryptocurrencies”. In: *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*. 2019, pp. 3–20.

[42] M. Brengel and C. Rossow. “Identifying Key Leakage of Bitcoin Users”. In: *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*. 2018, pp. 623–643.

[43] M. Buser et al. “A Survey on Exotic Signatures for Post-Quantum Blockchain: Challenges and Research Directions”. In: *ACM Comput. Surv.* 12 (2023).

[44] M. Caldwell and A. Voisine. *BIP38 Proposal*. `https://en.bitcoin.it/wiki/BIP_0038`. 2012.

[45] J. Camenisch, A. Lehmann, G. Neven, and K. Samelin. “Virtual Smart Cards: How to Sign with a Password and a Server”. In: *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*. 2016, pp. 353–371.

[46] R. Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 2001, pp. 136–145.

[47] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. “UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts”. In: *CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. 2020, pp. 1769–1787.

[48] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. “Bandwidth-Efficient Threshold EC-DSA”. In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*. 2020, pp. 266–296.

[49] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. “Bandwidth-efficient threshold EC-DSA revisited: Online/Offline Extensions, Identifiable Aborts, Proactivity and Adaptive Security”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 291.

[50] Certik. *2022 Year in Review - Crypto Wallet Security Incidents.* `https://www.certik.com/resources/blog/01iz10lvnaAIcuNZ2zNJqA-2022-year-in-review-crypto-wallet-security-incidents`. 2023.

[51] D. Chaum. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.* 1982, pp. 199–203.

[52] D. Chaum. "Security Without Identification: Transaction Systems to Make Big Brother Obsolete". In: *Commun. ACM* 10 (1985), pp. 1030–1044.

[53] D. Chaum, C. Crépeau, and I. Damgård. "Multiparty Unconditionally Secure Protocols (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA.* 1988, pp. 11–19.

[54] D. Chaum, A. Fiat, and M. Naor. "Untraceable Electronic Cash". In: *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings.* 1988, pp. 319–327.

[55] D. Chaum, M. Larangeira, and M. Yaksetig. "WOTSwana: A Generalized Sleeve Construction for Multiple Proofs of Ownership". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1623.

[56] D. Chaum, M. Larangeira, M. Yaksetig, and W. Carter. "W-OTS$^+$ Up My Sleeve! A Hidden Secure Fallback for Cryptocurrency Wallets". In: *Applied Cryptography and Network Security - 19th International Conference, ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings, Part I.* 2021, pp. 195–219.

[57] L. Chen, C. Kudla, and K. G. Paterson. "Concurrent Signatures". In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings.* 2004, pp. 287–305.

[58] C. Chuang, I. Hsu, and T. Lee. "A Two-Party Hierarchical Deterministic Wallets in Practice". In: *Proceedings of the 20th International Conference on Security and Cryptography, SECRYPT 2023, Rome, Italy, July 10-12, 2023.* 2023, pp. 850–856.

[59] Cointelegraph. *Slope wallets blamed for Solana-based wallet attack.* `https://cointelegraph.com/news/slope-wallets-blamed-for-solana-based-wallet-attack`. 2022.

[60] A. Cojocaru, J. A. Garay, A. Kiayias, F. Song, and P. Wallden. "Post-Quantum Security of the Bitcoin Backbone and Quantum Multi-Solution Bernoulli Search". In: *CoRR* (2020). arXiv: `2012.15254`.

[61] J. Coron. “Optimal Security Proofs for PSS and Other Signature Schemes”. In: *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings.* 2002, pp. 272–287.

[62] N. T. Courtois, P. Emirdag, and F. Valsorda. “Private Key Recovery Combination Attacks: On Extreme Fragility of Popular Bitcoin Key Management, Wallet and Cold Storage Solutions in Presence of Poor RNG Events”. In: *IACR Cryptol. ePrint Arch.* (2014), p. 848.

[63] W. Dai, T. Okamoto, and G. Yamamoto. “Stronger Security and Generic Constructions for Adaptor Signatures”. In: *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings.* 2022, pp. 52–77.

[64] A. P. K. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Shulman. “Securing DNSSEC Keys via Threshold ECDSA from Generic MPC”. In: *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II.* 2020, pp. 654–673.

[65] I. Damgård. *On Σ-Protocols.* `https://www.cs.au.dk/~ivan/Sigma.pdf`. 2010.

[66] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “BIP32-Compatible Threshold Wallets”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 312. **Part of this thesis.**

[67] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “The Exact Security of BIP32 Wallets”. In: *CCS ’21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021.* 2021, pp. 1020–1042. **Part of this thesis.**

[68] P. Das, S. Faust, and J. Loss. “A Formal Treatment of Deterministic Wallets”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.* 2019, pp. 651–668.

[69] Y. Deng, S. Ma, X. Zhang, H. Wang, X. Song, and X. Xie. “Promise $\varSigma $-Protocol: How to Construct Efficient Threshold ECDSA from Encryptions Based on Class Groups”. In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV.* 2021, pp. 557–586.

[70] A. Deshpande and M. Herlihy. “Privacy-Preserving Cross-Chain Atomic Swaps”. In: *Financial Cryptography and Data Security - FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers.* 2020, pp. 540–549.

[71] C. DeVon. *Crypto investors lost nearly $4 billion to hackers in 2022.* https://www.cnbc.com/2023/02/04/crypto-investors-lost-nearly-4-billion-dollars-to-hackers-in-2022.html. 2023.

[72] J. R. Douceur. "The Sybil Attack". In: *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers.* 2002, pp. 251–260.

[73] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. "Lattice Signatures and Bimodal Gaussians". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I.* 2013, pp. 40–56.

[74] C. Dwork and M. Naor. "Pricing via Processing or Combatting Junk Mail". In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings.* 1992, pp. 139–147.

[75] *Electrum Wallet.* https://electrum.org/.

[76] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. "Two-Party Adaptor Signatures From Identification Schemes". In: *IACR Cryptol. ePrint Arch.* (2021), p. 150.

[77] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. "Two-Party Adaptor Signatures from Identification Schemes". In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I.* 2021, pp. 451–480. **Part of this thesis.**

[78] A. Erwig, S. Faust, and S. Riahi. "Large-Scale Non-Interactive Threshold Cryptosystems in the YOSO Model". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1290.

[79] A. Erwig, S. Faust, S. Riahi, and T. Stöckert. "CommiTEE: An Efficient and Secure Commit-Chain Protocol using TEEs". In: *IACR Cryptol. ePrint Arch.* (2020), 1486. To appear at IEEE EuroS&P 2023.

[80] A. Erwig, M. Fischlin, M. Hald, D. Helm, R. Kiel, F. Kübler, M. Kümmerlin, J. Laenge, and F. Rohrbach. "Redactable Graph Hashing, Revisited - (Extended Abstract)". In: *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II.* 2017, pp. 398–405.

[81] A. Erwig, J. Hesse, M. Orlt, and S. Riahi. "Fuzzy Asymmetric Password-Authenticated Key Exchange". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II.* 2020, pp. 761–784.

[82] A. Erwig and S. Riahi. “Deterministic Wallets for Adaptor Signatures”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 1450.

[83] A. Erwig and S. Riahi. “Deterministic Wallets for Adaptor Signatures”. In: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II.* 2022, pp. 487–506. **Part of this thesis.**

[84] M. F. Esgin, O. Ersoy, and Z. Erkin. “Post-Quantum Adaptor Signatures and Payment Channel Networks”. In: *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II.* 2020, pp. 378–397.

[85] M. F. Esgin, O. Ersoy, V. Kuchta, J. Loss, A. Sakzad, R. Steinfeld, W. Yang, and R. K. Zhao. “A New Look at Blockchain Leader Election: Simple, Efficient, Sustainable and Post-Quantum”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 993.

[86] M. F. Esgin, R. Steinfeld, and R. K. Zhao. “MatRiCT$^+$: More Efficient Post-Quantum Private Blockchain Payments”. In: *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022.* 2022, pp. 1281–1298.

[87] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. “MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.* 2019, pp. 567–584.

[88] I. Eyal. “On Cryptocurrency Wallet Design”. In: *3rd International Conference on Blockchain Economics, Security and Protocols, Tokenomics 2021, November 18-19, 2021, New York University, USA (Virtual Conference).* 2021, 4:1–4:16.

[89] C. Fan, Y. Tseng, H. Su, R. Hsu, and H. Kikuchi. “Secure Hierarchical Bitcoin Wallet Scheme Against Privilege Escalation Attacks”. In: *IEEE Conference on Dependable and Secure Computing, DSC 2018, Kaohsiung, Taiwan, December 10-13, 2018.* 2018, pp. 1–8.

[90] L. D. Feo and S. D. Galbraith. “SeaSign: Compact Isogeny Signatures from Class Group Actions”. In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III.* 2019, pp. 759–789.

[91] L. D. Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. “SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies”. In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I.* 2020, pp. 64–93.

[92] M. Fersch, E. Kiltz, and B. Poettering. “On the One-Per-Message Unforgeability of (EC)DSA and Its Variants”. In: *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II.* 2017, pp. 519–534.

[93] A. Fiat and A. Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology - CRYPTO ’86, Santa Barbara, California, USA, 1986, Proceedings.* 1986, pp. 186–194.

[94] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. “Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys”. In: *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I.* 2016, pp. 301–330.

[95] P.-A. Fouque et al. “Falcon: Fast-Fourier lattice-based compact signatures over NTRU”. In: *Submission to the NIST’s post-quantum cryptography standardization process* 5 (2018).

[96] L. Fournier. *One-Time Verifiably Encrypted Signatures A.K.A. Adaptor Signatures.* `https://tinyurl.com/y4qxopxp`. 2019.

[97] D. Galindo, J. Liu, M. Ordean, and J. Wong. “Fully Distributed Verifiable Random Functions and their Application to Decentralised Random Beacons”. In: *IEEE European Symposium on Security and Privacy, EuroS&P 2021, Vienna, Austria, September 6-10, 2021.* 2021, pp. 88–102.

[98] S. Gao, T. Zheng, Y. Guo, and B. Xiao. “Efficient and Post-Quantum Zero-Knowledge Proofs for Blockchain Confidential Transaction Protocols”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1674.

[99] S. Garg, C. Gentry, A. Sahai, and B. Waters. “Witness encryption and its applications”. In: *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013.* 2013, pp. 467–476.

[100] R. Gennaro and S. Goldfeder. “Fast Multiparty Threshold ECDSA with Fast Trustless Setup”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018.* 2018, pp. 1179–1194.

[101] R. Gennaro and S. Goldfeder. “One Round Threshold ECDSA with Identifiable Abort”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 540.

[102] M. Gentilal, P. Martins, and L. Sousa. “TrustZone-backed bitcoin wallet”. In: *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC 2017, Stockholm, Sweden, January 24, 2017.* 2017, pp. 25–28.

[103] C. Gentry, C. Peikert, and V. Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008.* 2008, pp. 197–206.

[104] V. Gilchrist. *An Isogeny-Based Adaptor Signature Using SQISign.* `https://uwspace.uwaterloo.ca/bitstream/handle/10012/18157/Gilchrist_Valerie.pdf?sequence=3&isAllowed=y`. 2022.

[105] O. Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge, UK, 2004.

[106] O. Goldreich, S. Micali, and A. Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA.* 1987, pp. 218–229.

[107] S. Goldwasser and R. Ostrovsky. "Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent (Extended Abstract)". In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings.* 1992, pp. 228–245.

[108] J. Groth and V. Shoup. "Design and analysis of a distributed ECDSA signing service". In: *IACR Cryptol. ePrint Arch.* (2022), p. 506.

[109] J. Groth and V. Shoup. "On the Security of ECDSA with Additive Key Derivation and Presignatures". In: *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I.* 2022, pp. 365–396.

[110] L. C. Guillou and J. Quisquater. "A "Paradoxical" Indentity-Based Signature Scheme Resulting from Zero-Knowledge". In: *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings.* 1988, pp. 216–231.

[111] G. Gutoski and D. Stebila. "Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage". In: *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers.* 2015, pp. 497–504.

[112] L. Hanzlik, J. Loss, S. A. K. Thyagarajan, and B. Wagner. "Sweep-UC: Swapping Coins Privately". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1605.

[113] L. Hanzlik, J. Loss, and B. Wagner. "Token meets Wallet: Formalizing Privacy and Revocation for FIDO2". In: *IACR Cryptol. ePrint Arch.* (2022), p. 84.

[114] M. Hu. "Post-Quantum Secure Deterministic Wallet: Stateless, Hot/Cold Setting, and More Secure". In: *IACR Cryptol. ePrint Arch.* (2023), p. 62.

[115] K. Itakura. “A public-key cryptosystem suitable for digital multisignatures”. In: 1983.

[116] D. Johnson, A. Menezes, and S. A. Vanstone. “The Elliptic Curve Digital Signature Algorithm (ECDSA)”. In: *Int. J. Inf. Sec.* 1 (2001), pp. 36–63.

[117] S. A. Kakvi and E. Kiltz. “Optimal Security Proofs for Full Domain Hash, Revisited”. In: *J. Cryptol.* 1 (2018), pp. 276–306.

[118] J. Katz and N. Wang. “Efficiency improvements for signature schemes with tight security reductions”. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003.* 2003, pp. 155–164.

[119] E. Kiltz, J. Loss, and J. Pan. “Tightly-Secure Signatures from Five-Move Identification Protocols”. In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III.* 2017, pp. 68–94.

[120] E. Kiltz, V. Lyubashevsky, and C. Schaffner. “A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model”. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III.* 2018, pp. 552–586.

[121] E. Kiltz, D. Masny, and J. Pan. “Optimal Security Proofs for Signatures from Identification Schemes”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II.* 2016, pp. 33–61.

[122] J. B. Klamti and M. A. Hasan. “Post-Quantum Two-Party Adaptor Signature Based on Coding Theory”. In: *Cryptography* 1 (2022).

[123] Y. Kondi, B. Magri, C. Orlandi, and O. Shlomovits. “Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* 2021, pp. 608–625.

[124] D. Kravitz. *Digital Signature Algorithm.* U.S. patent #5,231,668. 1993.

[125] A. Langlois and D. Stehlé. “Worst-case to average-case reductions for module lattices”. In: *Des. Codes Cryptogr.* 3 (2015), pp. 565–599.

[126] D. Le, G. Yang, and A. A. Ghorbani. “DDH-based Multisignatures with Public Key Aggregation”. In: *IACR Cryptol. ePrint Arch.* (2019), p. 771.

[127] *Ledger Wallet.* `https://www.ledger.com/`.

[128] T. Lee, M. Ray, and M. Santha. “Strategies for Quantum Races”. In: *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. 2019, 51:1–51:21.

[129] Y. Lindell. “Fast Secure Two-Party ECDSA Signing”. In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*. 2017, pp. 613–644.

[130] Y. Lindell and A. Nof. “Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 2018, pp. 1837–1854.

[131] J. K. Liu, V. K. Wei, and D. S. Wong. “Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)”. In: *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*. 2004, pp. 325–335.

[132] A. D. Luzio, D. Francati, and G. Ateniese. “Arcula: A Secure Hierarchical Deterministic Wallet for Multi-asset Blockchains”. In: *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*. 2020, pp. 323–343.

[133] A. Lysyanskaya. “Unique Signatures and Verifiable Random Functions from the DH-DDH Separation”. In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. 2002, pp. 597–612.

[134] V. Lyubashevsky. “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures”. In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*. 2009, pp. 598–616.

[135] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi. “Concurrency and Privacy with Payment-Channel Networks”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 2017, pp. 455–471.

[136] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei. “Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability”. In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. 2019.

[137] E. V. Mangipudi, U. Desai, M. Minaei, M. Mondal, and A. Kate. “Uncovering Impact of Mental Models towards Adoption of Multi-device Crypto-Wallets”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 75.

[138] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song. "CHURP: Dynamic-Committee Proactive Secret Sharing". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.* 2019, pp. 2369–2386.

[139] A. Marcedone, R. Pass, and A. Shelat. "Minimizing Trust in Hardware Wallets with Two Factor Signatures". In: *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers.* 2019, pp. 407–425.

[140] G. Maxwell. *Deterministic Wallets.* `https://bitcointalk.org/index.php?topic=19137.msg239768`. 2011.

[141] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. "Simple Schnorr multi-signatures with applications to Bitcoin". In: *Des. Codes Cryptogr.* 9 (2019), pp. 2139–2164.

[142] S. Micali, K. Ohta, and L. Reyzin. "Accountable-subgroup multisignatures: extended abstract". In: *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.* 2001, pp. 245–254.

[143] S. Micali and A. Shamir. "An Improvement of the Fiat-Shamir Identification and Signature Scheme". In: *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings.* 1988, pp. 244–247.

[144] D. Micciancio and O. Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *SIAM J. Comput.* 1 (2007), pp. 267–302.

[145] P. Moreno-Sanchez and A. Kate. *Scriptless Scripts with ECDSA.* `https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf`. 2018.

[146] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* 2008.

[147] V. I. Nechaev. "Complexity of a Determinate Algorithm for the Discrete Logarithm". In: *Mathematical Notes* 2 (1994), pp. 165–172.

[148] Nikolaos Makriyannis and Oren Yomtov. *Practical Key-Extraction Attacks in Leading MPC Wallets.* `https://github.com/fireblocks-labs/mpc-ecdsa-attacks-23/blob/main/WhitePaper.pdf`. 2023.

[149] Nikolaos Makriyannis and Udi Peled. *A Note on the Security of GG18.* `https://info.fireblocks.com/hubfs/A_Note_on_the_Security_of_GG.pdf`. 2021.

[150] S. Noether. "Ring SIgnature Confidential Transactions for Monero". In: *IACR Cryptol. ePrint Arch.* (2015), p. 1098.

[151] K. Ohta and T. Okamoto. “A Digital Multisignature Scheme Based on the Fiat-Shamir Scheme”. In: *Advances in Cryptology - ASIACRYPT ’91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*. 1991, pp. 139–148.

[152] A. Poelstra. *Scriptless Scripts*. `https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf`. 2017.

[153] B. Poettering and D. Stebila. “Double-Authentication-Preventing Signatures”. In: *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I*. 2014, pp. 436–453.

[154] X. Qin et al. “BlindHub: Bitcoin-Compatible Privacy-Preserving Payment Channel Hubs Supporting Variable Amounts”. In: *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA, 2023, pp. 2020–2038.

[155] *Quantum Resistant Ledger (QRL)*. `https://github.com/theQRL/Whitepaper/blob/master/QRL_whitepaper.pdf`.

[156] R. L. Rivest, A. Shamir, and Y. Tauman. “How to Leak a Secret”. In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. 2001, pp. 552–565.

[157] T. Ruffing, S. A. K. Thyagarajan, V. Ronge, and D. Schröder. “(Short Paper) Burning Zerocoins for Fun and for Profit - A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol”. In: *Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018*. 2018, pp. 116–119.

[158] N. van Saberhagen. *CryptoNote v 2.0*. `https://bytecoin.org/old/whitepaper.pdf`. 2013.

[159] O. Sattath. “On the insecurity of quantum Bitcoin mining”. In: *Int. J. Inf. Sec.* 3 (2020), pp. 291–302.

[160] C. Schnorr. “Efficient Identification and Signatures for Smart Cards”. In: *Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. 1989, pp. 239–252.

[161] D. A. Schultz, B. Liskov, and M. D. Liskov. “MPSS: Mobile Proactive Secret Sharing”. In: *ACM Trans. Inf. Syst. Secur.* 4 (2010), 34:1–34:32.

[162] S. Shen, A. Rezapour, and W. Tzeng. “Unique Signature with Short Output from CDH Assumption”. In: *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*. 2015, pp. 475–488.

[163] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994.* 1994, pp. 124–134.

[164] V. Shoup. “Lower Bounds for Discrete Logarithms and Related Problems”. In: *Advances in Cryptology - EUROCRYPT ’97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding.* 1997, pp. 256–266.

[165] V. Shoup. “Sequences of games: a tool for taming complexity in security proofs”. In: *IACR Cryptol. ePrint Arch.* (2004), p. 332.

[166] Z. Sui, J. K. Liu, J. Yu, and X. Qin. “MoNet: A Fast Payment Channel Network for Scriptless Cryptocurrency Monero”. In: *42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022.* 2022, pp. 280–290.

[167] E. Tairi, P. Moreno-Sanchez, and M. Maffei. “A^{2}L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* 2021, pp. 1834–1851.

[168] E. Tairi, P. Moreno-Sanchez, and M. Maffei. “Post-Quantum Adaptor Signature for Privacy-Preserving Off-Chain Payments”. In: *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II.* 2021, pp. 131–150.

[169] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder. “Verifiable Timed Signatures Made Practical”. In: *CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.* 2020, pp. 1733–1750.

[170] S. A. K. Thyagarajan and G. Malavolta. “Lockable Signatures for Blockchains: Scriptless Scripts for All Signatures”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021.* 2021, pp. 937–954.

[171] S. A. K. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez. “Universal Atomic Swaps: Secure Exchange of Coins Across All Blockchains”. In: *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022.* 2022, pp. 1299–1316.

[172] N. Y. Times. *Cryptocurrency Hardware Wallets Can Get Hacked Too.* `https://www.wired.com/story/cryptocurrency-hardware-wallets-can-get-hacked-too/`. 2020.

[173] W. A. A. Torres, V. Kuchta, R. Steinfeld, A. Sakzad, J. K. Liu, and J. Cheng. "Lattice RingCT V2.0 with Multiple Input and Multiple Output Wallets". In: *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings.* 2019, pp. 156–175.

[174] W. A. A. Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng. "Post-Quantum One-Time Linkable Ring Signature and Application to Ring Confidential Transactions in Blockchain (Lattice RingCT v1.0)". In: *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings.* 2018, pp. 558–576.

[175] *Trezor Wallet.* `https://trezor.io/`.

[176] B. Tu, M. Zhang, and C. Yu. "Efficient ECDSA-Based Adaptor Signature for Batched Atomic Swaps". In: *Information Security - 25th International Conference, ISC 2022, Bali, Indonesia, December 18-22, 2022, Proceedings.* 2022, pp. 175–193.

[177] M. Turuani, T. Voegtlin, and M. Rusinowitch. "Automated Verification of Electrum Wallet". In: *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers.* 2016, pp. 27–42.

[178] D. Unruh. "Revocable Quantum Timed-Release Encryption". In: *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings.* 2014, pp. 129–146.

[179] P. Wuille. *BIP32 Proposal.* `https://en.bitcoin.it/wiki/BIP_0032`. 2012.

[180] Yehuda Lindell. *Cryptography and MPC in Coinbase Wallet as a Service (WaaS).* `https://coinbase.bynder.com/m/687ea39fd77aa80e/original/CB-MPC-Whitepaper.pdf`. 2023.

[181] X. Yin, Z. Liu, G. Yang, G. Chen, and H. Zhu. "Secure Hierarchical Deterministic Wallet Supporting Stealth Address". In: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part I.* 2022, pp. 89–109.

[182] M. Zhandry. "How to Construct Quantum Random Functions". In: *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012.* 2012, pp. 679–687.

[183] H. Zhang, F. Zhang, H. Tian, and M. H. Au. "Anonymous Post-Quantum Cryptocash". In: *Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers.* 2018, pp. 461–479.

[184] D. Zindros. “Hours of Horus: Keyless Cryptocurrency Wallets”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 715.

[185] G. Zyskind, A. Yanai, and A. S. Pentland. *Unstoppable Wallets: Chain-assisted Threshold ECDSA and its Applications.* Cryptology ePrint Archive, Paper 2023/832. `https://eprint.iacr.org/2023/832`. 2023.

List of Figures

List of Abbreviations

BIP32	Bitcoin Improvement Proposal 32
DDH	Decisional Diffie-Hellman
dlog	Discrete Logarithm
ECDSA	Elliptic Curve Digital Signature Algorithm
ID	Identification
KOSK	Knowledge of Secret Key
MLWE	Module Learning With Errors
MPC	Multi-Party Computation
NIST	National Institute of Standards and Technology
PIN	Personal Identification Number
PoW	Proof-of-Work
PPT	Probabilistic Polynomial-Time
QROM	Quantum Random Oracle Model
RingCT	Ring Confidential Transaction
RKA	Related Key Attack
SHA-512	Secure Hash Algorithm 512
TVRF	Threshold Verifiable Random Function
UC	Universal Composability

A. The Exact Security of BIP32 Wallets

In this chapter, we present the following publication with minor changes:

[67] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. "The Exact Security of BIP32 Wallets". In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021.* 2021, pp. 1020–1042. **Part of this thesis.**

The Exact Security of BIP32 Wallets

Poulami Das [1] Andreas Erwig [1] Sebastian Faust [1] Julian Loss [2]
Siavash Riahi [1]

[1] TU Darmstadt, Germany
`firstname.lastname@tu-darmstadt.de`
[2] University of Maryland, USA
`lossjulian@gmail.com`

Abstract

In many cryptocurrencies, the problem of key management has become one of the most fundamental security challenges. Typically, keys are kept in designated schemes called *wallets*, whose main purpose is to store these keys securely. One such system is the BIP32 wallet (Bitcoin Improvement Proposal 32), which since its introduction in 2012 has been adopted by countless Bitcoin users and is one of the most frequently used wallet system today. Surprisingly, very little is known about the concrete security properties offered by this system. In this work, we propose the first formal analysis of the BIP32 system in its entirety and without any modification. Building on the recent work of Das et al. (CCS '19), we put forth a formal model for hierarchical deterministic wallet systems (such as BIP32) and give a security reduction in this model from the existential unforgeability of the ECDSA signature algorithm that is used in BIP32. We conclude by giving concrete security parameter estimates achieved by the BIP32 standard, and show that by moving to an alternative key derivation method we can achieve a tighter reduction offering an additional 20 bits of security (111 vs. 91 bits of security) at no additional costs.

Keywords: Wallets, cryptocurrencies, foundations, BIP32

1 Introduction

Decentralized cryptocurrencies such as Bitcoin or Ethereum have introduced a new digital payment paradigm which does not rely on a central authority such as a bank or a credit card company. The main building block used in many popular cryptocurrencies to facilitate secure transfer and holding of assets are digital signatures. Loosely speaking, a user Alice in the system is identified by her public key pk_A which she uses as her address for receiving and sending payments. If Alice wants to send c coins of the underlying currency to another user Bob with address pk_B, she creates a transaction tx saying "Send c coins from pk_A to pk_B" and signs tx using her secret key sk_A. She then uploads the transaction tx together with the signature σ to the public ledger (often also called blockchain) of the cryptocurrency. Once the tuple (tx, σ) is visible on the public ledger, the payment is completed meaning that now Bob owns an additional c coins of the underlying currency. Clearly, Alice's funds remain secure only as long as no one can forge a signature σ on her behalf that verifies under pk_A. On top of this, it is generally recommended to use a fresh signing key for every new transaction stored on the public ledger to avoid that all transactions are linkable to the same user Alice. In the cryptocurrency space, the management and storage of secret keys is typically carried out by so-called *wallets* – which are pivotal for the security of cryptocurrency funds. Indeed, cryptocurrency wallets are a highly attractive target for hackers as illustrated by spectacular attacks against common cryptocurrency projects. For example, in 2018 alone, hackers managed to steal more than one billion USD worth of cryptocurrency from wallets [Ske18, Blo18, Bit18].

While several recent works study the formal security properties of cryptocurrency wallets (see related work for a detailed discussion), one of the most widely used schemes – *the BIP32 wallet* [Wik18] – has not been formally analyzed so far. This is somewhat surprising as BIP32 became a standard for deterministic Bitcoin wallets in 2012, and has been widely adopted since then (e.g., it is used in the deployment of popular wallets [Ele13, Tre14, Led14]). In this work, we address this gap and provide the first comprehensive study of the security properties achieved by the BIP32 wallet standard.

1.1 Deterministic Wallets

As we have already pointed out, to improve privacy it is important to not re-use the same signing key for too many public transactions. To explain why privacy is also beneficial for security, let us consider a user Alice who holds a single secret/public key pair $(\mathsf{sk}_A, \mathsf{pk}_A)$, and that she receives multiple payments to her address pk_A. As we have explained, such transactions contain her public key pk_A and are posted to the public ledger. Hence, an attacker can easily extract Alice's balance via the public transaction ledger. Over time, pk_A's balance might grow, and at some point, the attacker may identify pk_A as a high-priority target. The obvious approach to thwart an attacker's attempts of linking Alice's transactions would be for Alice to keep a set of l one-time random key pairs $\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_l, \mathsf{pk}_l)\}$ within her wallet, where each key pair is used for a single transaction on the public ledger. However, this approach has the obvious downside of Alice having to store all of her keys on disk (as long as they still retain some amount of currency). This requires a lot of storage space and bears the risk of losing one of her keys, at which point the associated funds of that key are irrevocably lost. A simple approach to overcome these issues are *deterministic wallets*, proposed by Buterin [But13]. A deterministic wallet usually contains a pair of master keys $(\mathsf{msk}, \mathsf{mpk})$ and a seed ch, which is also referred to as the *chaincode*. For every new transaction, the wallet deterministically derives a fresh session key pair $(\mathsf{sk}, \mathsf{pk})$ from the master keys with the help of deterministic key derivation algorithms. More precisely, the public key derivation algorithm takes as input the master public key mpk, the chaincode ch and an identifier ID and deterministically computes a one-time public key $\mathsf{pk}_{\mathsf{ID}}$. An analogous secret key derivation algorithm takes in msk, ch, and ID and deterministically computes a one-time secret key $\mathsf{sk}_{\mathsf{ID}}$ that matches $\mathsf{pk}_{\mathsf{ID}}$ (given that the arguments ch and ID in both derivations are identical). Going back to the example of BIP32, $(\mathsf{msk}, \mathsf{mpk})$ are generated as ECDSA keys and public key derivation is done by computing the offset $\omega := \mathsf{H}(\mathsf{mpk}, \mathsf{ch}, \mathsf{ID})$, where $\mathsf{ID} \in [2^{32}]$ and then rerandomizing mpk to $\mathsf{pk}_{\mathsf{ID}}$ by computing $\mathsf{pk}_{\mathsf{ID}} := \mathsf{mpk} + G \cdot \omega$. Here, G denotes the base point of an elliptic curve group of prime order p. A matching secret key can be derived via $\mathsf{sk}_{\mathsf{ID}} := \mathsf{msk} + \omega \pmod p$.

Hot/Cold Wallets. A typical way of using deterministic wallets in practice is via the hot/cold wallet paradigm. With this approach, Alice maintains two wallets. The first wallet is referred to as the *cold wallet*. It keeps the master secret key msk as well as the chaincode. The cold wallet is usually implemented via some simple storage device that should be almost permanently disconnected from the internet, so as to minimize the risk of attack. The second wallet is the so-called *hot wallet*, which is permanently online and keeps the master public key as well as the chaincode. Using the deterministic key derivation procedures, the two wallets can independently derive matching keys to use for one-time transaction on the public ledger. In a bit more detail, Alice uses her hot wallet as a low-security spending wallet which, at any point in time, keeps only a small amount of currency. Whenever the funds stored on the hot wallet exceed a certain amount, Alice can use the public key derivation algorithm to derive a new public key $\mathsf{pk}_{\mathsf{ID}}$ on her hot wallet and transfer the excess funds to $\mathsf{pk}_{\mathsf{ID}}$. Note that this requires no interaction with the cold wallet. At a later point in time, the cold wallet can come online for a brief moment and spend the funds from $\mathsf{pk}_{\mathsf{ID}}$, using a matching secret key $\mathsf{sk}_{\mathsf{ID}}$ derived via the secret key derivation algorithm. As far as security goes, we would like to ensure two properties. First, *unlinkability* ensures that keys derived from the same master key pair are indistinguishable from random keys, given that the hot wallet has not leaked the chaincode to the attacker. Second, *unforgeability* ensures that even if the hot wallet leaks the chaincode (e.g., because it has been corrupted), signatures from derived keys should still remain unforgeable. While unlinkability is easy to achieve, unforgeability is a much more subtle issue in this setting, as the derived keys are all correlated once the chaincode has been revealed to the attacker. Hence, the standard unforgeability property of the underlying signature scheme is no longer sufficient to ensure unforgeability of signatures under these derived keys.

1.2 Limitations of Existing Works

The work of Das et al. [DFL19] was the first to provide a formal model to reason about the aforementioned security properties. It also showed how to achieve secure constructions in their proposed model from various different signature schemes used in practice, e.g., Schnorr, BLS, and ECDSA. Notably, the latter construction is very practical and can be integrated directly with the (unmodified) Bitcoin system. In spite of these achievements, their work makes no progress towards formally proving security properties for the BIP32 wallet standard that is widely used in many real-world systems. Let us discuss the reasons for this in a little more detail.

First, the construction of Das et al. uses a *multiplicative rerandomization* to derive keys, in which keys for identity ID are computed from $\omega = \mathsf{H}(\mathsf{mpk}, \mathsf{ch}, \mathsf{ID})$ as $\mathsf{pk}_{\mathsf{ID}} := \mathsf{mpk} \cdot \omega$, and $\mathsf{sk}_{\mathsf{ID}} := \mathsf{msk} \cdot \omega \pmod p$. By comparison, as we saw above, BIP32 uses an *additive rerandomization.* Although this might look like a minor difference, we will see later that the proof technique and security guarantees achieved by the additive version differ significantly from the multiplicative one. Second, the work of Das et al. does not consider the hierarchical key derivation mechanism provided by BIP32. Hierarchical deterministic wallets allow for keys in the wallet to act simultaneously as signing keys *and* as parent (master) keys to derive new child keys in their own right. As a useful example, consider a company that wishes to delegate new signing key pairs to different entities within the company. Unfortunately, it cannot be guaranteed that all entities in the company store their keys securely and some of them might be leaked to the adversary over time. Such a strong adversary cannot be captured by the model and constructions of Das et. al. Since many wallets that are used in practice follow the BIP32 standard, it is crucial to provide a formal analysis of the scheme *as is*, meaning without any modifications to it.

1.3 Our Contributions

In this work we address the above shortcomings and provide, for the first time, a formal analysis of the full BIP32 specification in the hot/cold wallet setting. An important implication of our work is that we can establish the exact security that is achieved by the current standard, which also leads us to propose a minor modification that can significantly improve security without any additional costs.

Rerandomizing ECDSA. We begin by recalling the notion of *unforgeability under honestly rerandomized keys (UFCMA-HRK)* introduced by Das et al. [DFL19]. As this notion will serve as the basis of our wallet constructions, we review it in detail below. Compared to the standard notion of unforgeability under chosen message attacks (UFCMA), the adversary in the UFCMA-HRK game initially obtains a challenge public key pk and gets to query for rerandomizations of pk. The game returns the rerandomized public key $\tilde{\mathsf{pk}}$ together with the (uniformly chosen) randomness ρ that was used in the rerandomization process. The exact way that the rerandomization is actually done depends on the scheme; we are mostly interested in the case where ECDSA keys are additively rerandomized as $\mathsf{pk} + G \cdot \rho$. The game then allows the adversary to query for signatures relative to any of the rerandomized public keys that it has previously obtained from the game. It is considered successful if it can return a forgery relative to any of the requested keys $\tilde{pk}$ on a message for which it has not previously asked for a signature under $\tilde{pk}$. As observed by Das et al., this security notion is a weakened version of *unforgeability under rerandomized keys* [FKM+16] in which the adversary can choose the random coins ρ itself and provide them to the game. In Section 3, we prove that ECDSA with additive rerandomization satisfies UFCMA-HRK as long as *each message is signed only once per key.* A first attempt is to naively follow the approach of Das et al. who showed that ECDSA with multiplicative rerandomization satisfies UFCMA-HRK (without any restrictions on the number of signatures per message). The main idea of Das et al.'s reduction from UFCMA-HRK to UFCMA (both with respect to the ECDSA scheme) is to rely on a *related key attack* (RKA) that is present in the multiplicatively rerandomized version of the ECDSA scheme. Concretely, the RKA allows to transform a signature (r, s) on message m_0 relative to a key pk_0 into a signature $(r, s/\rho)$ on message m_1 that is valid under the related key $\mathsf{pk}_1 = \mathsf{pk}_0 \cdot \rho$, where ρ satisfies $\rho = \frac{\mathsf{H}(m_0)}{\mathsf{H}(m_1)}$. This attack can be leveraged by the reduction to answer all signing queries in the UFCMA-HRK game. More precisely, using the RKA, it is possible to transform signatures obtained from the signing oracle in the UFCMA game into signatures relative to any of the rerandomized keys in the UFCMA-HRK game (via programming of the random oracle). Hence, we are immediately faced with the following obstacle: this RKA does not work if keys are additively rerandomized.

Extending to Additive Rerandomization. To overcome this issue with the existing reduction, we present a new RKA which works for additively rerandomized ECDSA. The attack works as follows: given a signature (r, s) on m_0 relative to pk_0, (r, s) is also a valid signature relative to the public key $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$ on message m_1, given that $\rho = (\mathsf{H}(m_0) - \mathsf{H}(m_1))/r$. Rather surprisingly, considering ECDSA's huge popularity, we are not aware of this attack having been noticed previously. Using our new RKA, we are now able to (almost) make the simulation of signatures in Das et al.'s approach work. However, there is a further issue that comes from the structure of the additive RKA. Suppose that the reduction is directed to program the random oracle H on a message m so as to provide the attacker with a signature relative to a (rerandomized) public key $\tilde{\mathsf{pk}}$ in the UFCMA-HRK game. The above RKA forces the reduction to program H on a value that depends on a *particular signature* (r, s) on m, which it obtains from the signing oracle in the underlying UFCMA game. Now, the only signature on m that the reduction can hand to the adversary under pk is (r, s). If the adversary requests another signature *on the same message* m, we are not able to reply with a fresh signature, as we can program H on m only a single time. For this reason, we have to restrict ourselves to one-per-message unforgeability. We emphasize, however, that this notion of security (one signature per-message) is sufficient in our setting, as transactions are identified by unique nonces in most cryptocurrencies (including Bitcoin) and hence never signed twice. An additional benefit of our new reduction (compared to [DFL19]) is that it only requires the weaker assumption that the underlying ECDSA scheme is one signature per-message unforgeable in its own right. This is worth noting, as the work of Fersch et al. shows that ECDSA achieves this property in the random oracle model [FKP17] (albeit with a very large security loss). By comparison, the unrestricted security (i.e., UFCMA) of ECDSA remains only a conjecture in the plain random oracle model. Our reduction also removes the need for the random salt present in Das et al.'s construction. This is an important improvement, as it allows using BIP32 without Bitcoin's scripting language, which was required by the construction of Das et al. due to their use of the salt. Finally, we remark that our reduction (by comparison to Das et al.) is *non-tight* and loses a factor proportional to the total number of keys derived in the UFCMA-HRK game. We provide further discussion on this issue in the next section and in Section 3.

Hierarchical Wallets. To complete the analysis of BIP32, the second part of our work focuses on formal security properties when supporting hierarchies in deterministic wallet constructions (as is the case for BIP32). As already hinted, the core difficulty in this setting is that some of the wallet's keys may be given to untrustworthy users who may leak their cold wallet keys to the adversary. If this happens, it is important to ensure that the adversary does not gain information about secret keys further up in the hierarchy. It is easy to see that this property is not achieved if all keys are derived using the derivation algorithms described so far: if the adversary learns $\mathsf{sk}_{\mathsf{ID}} = \mathsf{msk} + \rho \pmod p$, where ρ is computed as $\rho = \mathsf{H}(\mathsf{mpk}, \mathsf{ch}, \mathsf{ID})$, then it can recover msk as $\mathsf{msk} = \mathsf{sk}_{\mathsf{ID}} - \rho \pmod p$ and learn all cold wallet keys that were ever derived using msk. Because of this, BIP32 offers a second mode of deriving keys called *hardened key derivation*. Hardened keys are derived by changing the computation of the offset ρ above to $\rho = \mathsf{H}(\mathsf{msk}, \mathsf{ch}, \mathsf{ID})$. Now, even when learning $\mathsf{sk}_{\mathsf{ID}}$, it is not possible for the adversary to recover msk. The downside of hardened key derivation is that the hot and cold wallet can no longer independently derive keys (as the hot wallet does not know msk). Thus, this mode of derivation is not intended for use in the hot/cold wallet paradigm, but simply to create keys with a higher degree of security. These keys can either be stored (efficiently) as part of the main wallet or handed to users in the system without any concern for other cold wallet keys. In Section 4, we state the syntactical definition and correctness properties of a hierarchical deterministic wallet. We then introduce a security model that supports both types of key derivations (hardened and non-hardened), as well as secret key leakage of hardened keys. We refer to this notion of security as WUFCMA. In Section 5, we provide a generic construction HDWal that transforms a signature scheme satisfying UFCMA-HRK into a hierarchical deterministic wallet with WUFCMA security.[1] In this way, we are able to complete the analysis of BIP32 by instantiating HDWal with ECDSA using additive rerandomization.

On the Tightness of Our Construction. A particular focus of our work is to analyze the tightness and concrete security achieved by our constructions, most notably BIP32. We have already mentioned that our reduction from UFCMA-HRK to UFCMA of the ECDSA scheme with additive rerandomization is non-tight. More precisely, it loses a factor proportional to the number of keys derived by the adversary in the UFCMA-HRK game. Thus, our goal is to at least achieve the best possible tightness of our generic transform HDWal. To this end, let us first consider the possible options for potential security losses. From worst to best (excluding a tight reduction), the options are:

- Loss in the number of random oracle queries.
- Loss in the number of keys derived in the wallet (hardened or non-hardened).
- Loss in the number of signing oracle queries (assuming keys are used only once).
- Loss in the number of hardened keys leaked to the adversary.

[1] In case the underlying signature scheme has the one signature per message restriction, then the resulting wallet scheme also does.

The first three possibilities are quite catastrophic as the number of random oracle queries, signing oracle queries, or keys derived in practice could be quite high. On the other hand, we expect the number of *leaked keys* to be only a small portion of all the keys in a given wallet (we use 1% as an estimate in our calculations). We are able to prove that HDWal indeed achieves a multiplicative security loss proportional to only the hardened keys leaked to the adversary over the course of the lifetime of the wallet. Furthermore, we show that any *generic transform* from UFCMA-RK (a stronger notion than what is used in our construction) to WUFCMA *must lose at least this factor.* Hence, our construction HDWal achieves the *best possible parameters.* To prove our results, we adapt the reduction/metareduction techniques introduced by Coron in his seminal work [Cor02]. Given that his results deal with the tightness of unique signatures (which is very different from our setting), this requires careful insight into his technique in order to adapt it to our model.

Concrete Security Parameters. We conclude by giving a discussion of the concrete security levels achieved by BIP32 and the multiplicative ECDSA scheme of Das et al., when plugged into HDWal. We find that BIP32 gives roughly 94 bits of security according to our theorems and conservative choices of parameters. We find that by comparison, the multiplicative version of Das et al. gives 114 bits of security with a similarly efficient scheme. (We remark that using the techniques introduced in our paper, we can also remove the salt in the multiplicatively rerandomizable ECDSA version of Das et al.). Given these insights, we strongly recommend that the Bitcoin community switch rerandomizations in BIP32 from additive to multiplicative, in particular since these changes essentially come for free.

1.4 Related Work

The most relevant previous work for us is by Das et al. [DFL19] as mentioned previously. However, there have been other works which try to formalize cryptographic wallets. The work of Gutoski and Stebila [GS15] proposes an alternative construction for hierarchical wallets where up to d session keys can leak without the master secret key being compromised under the one-more discrete-log assumption. However, their security model is weaker than our model (or the security model of Das et al. on which we base our work). More precisely, in their model, the adversary cannot query the game for signatures under uncompromised wallet keys. Furthermore, instead of the traditional security model where the adversary wins if she can forge a signature, the adversary's goal in their security definition is to extract the master secret/public key pair. Another more recent work is by Luzio et al. [LFA20] where the authors design a new hierarchical wallet scheme by using (deterministic) hierarchical key assignment schemes [ABFF09]. Unfortunately, their solution is not compatible with cryptocurrencies such as Bitcoin since their solution requires a more sophisticated (signature) verification algorithm, where a certificate associated with the user needs to be verified along with the signature.

Turuani et al. [TVR16] analyzed the Bitcoin Electrum wallet using automated verification in the Dolev-Yao model. However, many automated verification models only consider "idealized" building blocks, i.e., cryptographic building blocks that are perfectly secure. Consequently, this type of analysis excludes weaknesses such as related key attacks, which are of fundamental relevance in the setting of deterministic wallets.

Another line of work has considered the security of hardware wallets [MPs19, AGKK19] and implementation bugs in wallets (such as weak randomness) [CEV14, BR18, BH19]. Additionally, there have been several works with focus on the use of threshold ECDSA signatures [KMOS19, GGN16, LN18, DKLs18] and multi-signatures [BDN18] in (and outside of) wallet systems.

In a recent work, Alkadri et al. [ADE+20] have shown how to realize deterministic wallets that are post-quantum secure. To this end, they suitably adapt the model and techniques of Das et al. by considering an adversary with quantum computing power.

The concept of rerandomizable signature schemes was first introduced by Fleischhacker et al. [FKM+16] and later used by [DFL19, ADE+20] for their wallet schemes. In addition, related key attacks have been studied for signature schemes such as Schnorr [Sch90] in many previous works [FF13, KMP16, ZCC+15]. For ECDSA, Das et al. leveraged related key attacks to achieve a multiplicatively rerandomizable ECDSA scheme which they prove secure w.r.t. the security notion of unforgeability under honestly rerandomizable keys. Finally, Fersch et. al. [FKP16] provided the first security analysis of ECDSA in an idealized model.

2 Preliminaries

Notation. We use the notation $s \xleftarrow{\$} H$ to denote the uniform sampling of a variable s from the set H. For an integer l, $[l]$ denotes the set of integers $\{1, \cdots, l\}$. We use upper case letters to denote algorithms. For an algorithm A, we write $y \xleftarrow{\$} A(x)$ to denote the execution of a randomized algorithm A on input x that outputs y. We write $y \leftarrow B(x; \rho)$ to denote the execution of an algorithm B that, on input x and randomness ρ, outputs y. Note that in this notation, B *is deterministic.* We use the notation $y \in A(x)$ to denote that y is in the set of possible outputs of A on input x.

In order to simplify our notation and definitions, we assume that public parameters par have been securely generated and can be used throughout the paper as input to algorithms. We generally assume that, initially, boolean variables are set to false, integers are set to 0, lists are set to $\emptyset$, and undefined entries of lists are set to $\perp$. For strings $a, b \in \{0,1\}^*$, we write $a = (b, \cdot)$ if b is a prefix of a and likewise, we write $a \neq (b, \cdot)$ if a is not prefixed by b. We denote by κ the security parameter throughout the paper.

We use standard code-based security games [Sho04]. A *game* $\mathbf{G}$ is an interactive probability experiment between an *adversary* $\mathcal{A}$ and an (implicit) *challenger* which provides answers to oracle queries posed by $\mathcal{A}$. The output of $\mathbf{G}$ when interacting with adversary $\mathcal{A}$ is denoted as $\mathbf{G}^{\mathcal{A}}$. Finally, the randomness in any probability term of the form $\Pr[\mathbf{G}^{\mathcal{A}} = 1]$ is assumed to be over all the random coins in game $\mathbf{G}$.

2.1 Signature Schemes

We now recall the definition of signature schemes and that of signature schemes with perfectly rerandomizable keys from [DFL19].

Definition 2.1 (Signature Scheme). A *signature scheme* is a tuple of algorithms $\mathsf{Sig} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ which are defined as follows:

- $\mathsf{Sig.Gen}(\mathsf{par})$: The randomized *key generation* algorithm $\mathsf{Sig.Gen}$ takes as input public parameters par and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sig.Sign}(\mathsf{sk}, m)$: The (possibly) randomized *signing* algorithm $\mathsf{Sig.Sign}$ takes as input a secret key sk and a message m and outputs a signature σ.
- $\mathsf{Sig.Verify}(m, \mathsf{pk}, \sigma)$: The deterministic *verification* algorithm $\mathsf{Sig.Verify}$ takes as input a public key pk, a signature σ, and a message m. It outputs either 1 (accept) or 0 (reject).

A signature scheme Sig is *correct* if the following holds: For all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Sig.Gen}(\mathsf{par})$ and all $m \in \{0,1\}^*$ we have that

$$\Pr_{\sigma \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}, m)} [\mathsf{Sig.Verify}(\mathsf{pk}, \sigma, m) = 1] = 1.$$

Definition 2.2 (Signature Scheme with Perfectly Rerandomizable Keys). A *signature scheme with perfectly rerandomizable keys* is a tuple of algorithms $\mathsf{RSig} = (\mathsf{RSig.Gen}, \mathsf{RSig.Sign}, \mathsf{RSig.Verify}, \mathsf{RSig.RandSK}, \mathsf{RSig.RandPK})$ where $(\mathsf{RSig.Gen}, \mathsf{RSig.Sign}, \mathsf{RSig.Verify})$ are the standard algorithms of a signature scheme. Moreover, we assume that the public parameters par define a randomness space $\mathcal{R} := \mathcal{R}(\mathsf{par})$. Then the algorithms $\mathsf{RSig.RandSK}$ and $\mathsf{RSig.RandPK}$ are defined as follows:

- $\mathsf{RSig.RandSK}(\mathsf{sk}; \rho)$: The deterministic *secret key rerandomization algorithm* $\mathsf{RSig.RandSK}$ takes as input a secret key sk and randomness $\rho \in \mathcal{R}$ and outputs a rerandomized secret key sk'.
- $\mathsf{RSig.RandPK}(\mathsf{pk}; \rho)$: The deterministic *public key rerandomization algorithm* $\mathsf{RSig.RandPK}$ takes as input a public key pk and randomness $\rho \in \mathcal{R}$ and outputs a rerandomized public key pk'.

We make the convention that for the empty string ϵ, we have that $\mathsf{RSig.RandPK}(\mathsf{pk}; \epsilon) = \mathsf{pk}$ and $\mathsf{RSig.RandSK}(\mathsf{sk}; \epsilon) = \mathsf{sk}$.

We further require:

1. *(Perfect) rerandomizability of keys:* For all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{RSig.Gen}(\mathsf{par})$ and $\rho \xleftarrow{\$} \mathcal{R}$, the distributions of $(\mathsf{sk}', \mathsf{pk}')$ and $(\mathsf{sk}'', \mathsf{pk}'')$ are identical, where:
$$(\mathsf{sk}', \mathsf{pk}') \leftarrow (\mathsf{RSig.RandSK}(\mathsf{sk}; \rho), \mathsf{RSig.RandPK}(\mathsf{pk}; \rho)),$$
$$(\mathsf{sk}'', \mathsf{pk}'') \xleftarrow{\$} \mathsf{RSig.Gen}(\mathsf{par}).$$

2. *Correctness under rerandomized keys:* For all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{RSig.Gen}(\mathsf{par})$, for all $\rho \in \mathcal{R}$, and for all $m \in \{0,1\}^*$, the rerandomized keys $\mathsf{sk}' \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}; \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}; \rho)$ satisfy:
$$\Pr_{\sigma \xleftarrow{\$} \mathsf{RSig.Sign}(\mathsf{sk}', m)} \left[\mathsf{RSig.Verify}\left(\mathsf{pk}', \sigma, m\right) = 1\right] = 1.$$

Security notion uf-cma1. In this work, we use the security notion of *one-per message existential unforgeability under chosen message attacks* (**uf-cma1**) [FKP17] which is a slightly weaker variant of the standard notion of existential unforgeability under chosen message attacks (**uf-cma**) security. In contrast to standard **uf-cma**, in **uf-cma1**, the adversary is restricted to querying the signing oracle at most once for each message. We formalize the **uf-cma1** notion for a signature scheme Sig in the form of a game $\mathbf{uf\text{-}cma1}_{\mathsf{Sig}}$ as follows.

Game $\mathbf{uf\text{-}cma1}_{\mathsf{Sig}}$:

- **Setup Phase:** The challenger initiates a list as $\mathsf{SigList} \leftarrow \{\epsilon\}$ for storing messages and samples a pair of keys $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.Gen}(\mathsf{par})$. Then, $\mathcal{A}$ is run on input pk.
- **Online Phase:** $\mathcal{A}$ is given access to a signing oracle $\mathtt{Sign}$ which works as follows. On input a message m, if m was queried in a previous $\mathtt{Sign}$ query, i.e., if $m \in \mathsf{SigList}$, then $\perp$ is returned. Otherwise, $\mathtt{Sign}$ computes a signature on message m as $\sigma \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}, m)$. The message m is stored in the $\mathsf{SigList}$ and the signature σ is returned as the answer.
- **Output Phase:** Finally, $\mathcal{A}$ wins the game if it can provide a forgery σ^* on a message m^*, where (1) m^* is fresh, i.e., $m^* \notin \mathsf{SigList}$ and (2) σ^* is a valid forgery, i.e., $\mathsf{Sig.Verify}(\mathsf{pk}, \sigma^*, m^*) = 1$.

For an algorithm $\mathcal{A}$ we define $\mathcal{A}$'s advantage in the game $\mathbf{uf\text{-}cma1}_{\mathsf{Sig}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\mathbf{uf\text{-}cma1}_{\mathsf{Sig}}} = \Pr[\mathbf{uf\text{-}cma1}^{\mathcal{A}}_{\mathsf{Sig}} = 1]$.

Security notion uf-cma-hrk1. For signature schemes with perfectly rerandomizable keys, we introduce the notion of *one-per message existential unforgeability under honestly rerandomizable keys* (**uf-cma-hrk1**), which restricts the security notion of *existential unforgeability under honestly rerandomizable keys* (**uf-cma-hrk**) as introduced by Das et al. [DFL19]. In this security notion, the signing oracle cannot only return signatures under sk, but it can also return signatures that were produced with keys that represent *honest* rerandomizations of sk. The term *honest* indicates that the randomness for the rerandomization is chosen uniformly at random from $\mathcal{R}$ (by the game itself). Our security notion of **uf-cma-hrk1** restricts the notion of **uf-cma-hrk** in the sense that the signing oracle returns at most one signature for each randomness/message pair (ρ, m). We formally model the notion of **uf-cma-hrk1** for a rerandomizable signature scheme RSig in the form of a game $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ as follows.

Game $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$:

- **Setup Phase:** The challenger initializes two lists as $\mathsf{SigList} \leftarrow \{\epsilon\}$ and $\mathsf{RList} \leftarrow \{\epsilon\}$ and samples a pair of keys $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{RSig.Gen}(\mathsf{par})$. Then $\mathcal{A}$ is run on input pk.
- **Online Phase:**
  - $\mathcal{A}$ is given access to an oracle $\mathtt{Rand}$, which, upon a query, samples a fresh random value from $\mathcal{R}$ as $\rho \xleftarrow{\$} \mathcal{R}$, stores ρ in the list RList, and returns ρ.
  - $\mathcal{A}$ is given access to a signing oracle $\mathtt{RSign}$ which works as follows. On input a message m and a randomness ρ, if ρ was not obtained via a prior $\mathtt{Rand}$ query (i.e., $\rho \notin \mathsf{RList}$), then return $\perp$. Otherwise, derive a pair of keys rerandomized with the randomness ρ, as $\mathsf{sk}' \leftarrow \mathsf{RSig.SKDer}(\mathsf{sk}; \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RSig.PKDer}(\mathsf{pk}; \rho)$. If $(\mathsf{pk}', m) \in \mathsf{SigList}$ then return $\perp$. Otherwise, a signature is derived on message m under the secret key sk' as $\sigma \leftarrow \mathsf{RSig.Sign}(\mathsf{sk}', m)$. The tuple (pk', m) is stored in the $\mathsf{SigList}$ and the signature σ is returned as the answer.

- **Output Phase:** $\mathcal{A}$ wins if it returns a forgery σ^* together with a message m^* and a public key $\mathsf{pk}^* \leftarrow \mathsf{RSig.PKDer}(\mathsf{pk}; \rho^*)$,[2] s.t. following holds: (1) the randomness ρ^* has been derived via a `Rand` query, i.e., $\rho^* \in \mathsf{RList}$, (2) (m^*, ρ^*) is fresh, i.e., $(\mathsf{pk}^*, m^*) \notin \mathsf{SigList}$, and (3) σ^* is a valid forgery, i.e., $\mathsf{RSig.Verify}(\mathsf{pk}^*, \sigma^*, m^*) = 1$.

For an algorithm $\mathcal{A}$ we define $\mathcal{A}$'s advantage in game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} = \Pr[\textbf{uf-cma-hrk1}^{\mathcal{A}}_{\mathsf{RSig}} = 1]$.

Other than only allowing the adversary to ask for at most one signature per message, our definition deviates from the one presented in [DFL19] by storing the tuples (pk', m) in the list SigList instead of just storing m. This change allows an adversary in the **uf-cma-hrk1** game to query a signature for the same message but under different public keys.

3 Security Analysis of Additively Rerandomizable ECDSA

In the following discussion, let $\mathbb{E}(\mathsf{par})$ denote an elliptic curve with base point G and prime order p. Furthermore, assume hash functions $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ (modeled as random oracles). In this section, we present a signature scheme with rerandomizable keys $\mathsf{REC}[\mathsf{H}_1]$ based on the standard ECDSA scheme which we denote by $\mathsf{EC}[\mathsf{H}_0]$ (cf. Figure 1). $\mathsf{REC}[\mathsf{H}_1]$, as illustrated in Figure 7, works in a similar way as $\mathsf{EC}[\mathsf{H}_0]$ with two main differences. (1) It is extended by two algorithms RandSK and RandPK for the key rerandomization and (2) it is designed for key-prefixed messages. First, the two algorithms RandSK and RandPK randomize a key pair by *adding* a random value to each key. This is in contrast to the signature scheme with *multiplicatively* rerandomizable keys based on ECDSA as presented by Das et al. [DFL19], where the rerandomization algorithms multiply a random value to each key. Second, $\mathsf{REC}[\mathsf{H}_1]$ is designed for key-prefixed messages, i.e., upon executing $\mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}, m)$ for a secret key sk and a message m, the message is first extended to a *key-prefixed message* $\mathsf{pm} \leftarrow (\mathsf{pk}, m)$ where pk represents the public key corresponding to sk. Then the prefixed message pm is signed under sk.

We prove that $\mathsf{REC}[\mathsf{H}_1]$ satisfies **uf-cma-hrk1** security by providing a reduction from the **uf-cma1** security of the standard ECDSA scheme $\mathsf{EC}[\mathsf{H}_0]$. An integral part of the reduction is the observation that there exists a so-called "related key attack" (RKA) in the scheme $\mathsf{EC}[\mathsf{H}_0]$. An RKA allows to transform a signature that is valid under a public key pk_0 into a signature that is valid under another public key pk_1 given there exists a specific relation between pk_1 and pk_0. The RKA in $\mathsf{EC}[\mathsf{H}_0]$ allows to use a signature σ that is valid under a public key pk_0 as a valid signature under a public key pk_1 in case pk_1 and pk_0 are related as $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$, where ρ must satisfy $\rho = \frac{H_0(m_0) - H_1(m_1)}{r}$. We formally describe this related key attack in the following Lemma.

Lemma 3.1 *Let H_0, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ be hash functions (modeled as random oracles). Suppose that $\sigma = (r, s)$ is a valid signature on message $m_0 \in \{0,1\}^*$ w.r.t. $\mathsf{EC}[\mathsf{H}_0]$ and public key pk_0, i.e., $\mathsf{EC}[\mathsf{H}_0].\mathsf{Verify}(\mathsf{pk}_0, \sigma, m_0) = 1$. Furthermore, let $\rho = \frac{H_0(m_0) - H_1(m_1)}{r} \pmod p$. Then σ is also a valid signature on message $m_1 \in \{0,1\}^*$ w.r.t. $\mathsf{EC}[\mathsf{H}_1]$ and public key $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$, i.e., $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m_1) = 1$.*

Proof of Lemma 3.1. We have to show that $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m_1) = 1$ for $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$ and $\rho = \frac{H_0(m_0) - H_1(m_1)}{r} \pmod p$. Note that $\sigma = (r, s)$, where $s = t^{-1}(\mathsf{H}_0(m_0) + r\mathsf{sk}_0) \pmod p$ and r represents the x-coordinate of the elliptic curve point $t \cdot G$ for $t \xleftarrow{\$} \mathbb{Z}_p$. As shown in Figure 1, $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m_1)$ computes the following:

$$
\begin{aligned}
& u_1 \cdot G + u_2 \cdot \mathsf{pk}_1 \\
=& \mathsf{H}_1(m_1) \cdot s^{-1} \cdot G + r \cdot s^{-1} \cdot \left(\mathsf{pk}_0 + \frac{\mathsf{H}_0(m_0) - \mathsf{H}_1(m_1)}{r} \cdot G \right) \\
=& s^{-1} \cdot G \left(\mathsf{H}_1(m_1) + r \cdot \mathsf{sk}_0 + \mathsf{H}_0(m_0) - \mathsf{H}_1(m_1) \right) \\
=& s^{-1} \cdot G \left(r \cdot \mathsf{sk}_0 + \mathsf{H}_0(m_0) \right) \\
=& t \cdot (\mathsf{H}_0(m_0) + r\mathsf{sk}_0)^{-1} \cdot (\mathsf{H}_0(m_0) + r\mathsf{sk}_0) \cdot G = t \cdot G
\end{aligned}
$$

Since the x-coordinate of $t \cdot G$ equals $r \pmod p$, it holds that $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m_1) = 1$. ■

[2]For simplicity, we tacitly assume that pk^* identifies ρ^*. This can easily be achieved using appropriate bookkeeping.

```
Algorithm $\mathsf{EC}[\mathsf{H}_0].\mathsf{Gen}\,(\mathsf{par})$
00 $x \xleftarrow{\$} \mathbb{Z}_p$
01 $X \leftarrow x \cdot G$
02 $\mathsf{sk} \leftarrow x$
03 $\mathsf{pk} \leftarrow X$
04 Return $(\mathsf{pk}, \mathsf{sk})$

Algorithm
$\mathsf{EC}[\mathsf{H}_0].\mathsf{Sign}\,(\mathsf{sk} = x, m)$
05 $z \leftarrow \mathsf{H}_0(m)$
06 $t \xleftarrow{\$} \mathbb{Z}_p$
07 $(e_x, e_y) \leftarrow t \cdot G$
08 $r \leftarrow e_x \mod p$
09 If $r = 0 \mod p$
10    Goto Step 06
11 $s \leftarrow t^{-1}\,(z + rx) \mod p$
12 If $s = 0 \mod p$
13    Goto Step 06
14 Return $\sigma := (r, s)$

Algorithm
$\mathsf{EC}[\mathsf{H}_0].\mathsf{Verify}\,(\mathsf{pk} = X, \sigma, m)$
15 Parse $(r, s) \leftarrow \sigma$
16 If $(r, s) \notin \mathbb{Z}_p$
17    Return 0
18 $w \leftarrow s^{-1} \mod p$
19 $z \leftarrow \mathsf{H}_0(m)$
20 $u_1 \leftarrow zw \mod p$
21 $u_2 \leftarrow rw \mod p$
22 $(e_x, e_y) \leftarrow u_1 \cdot G + u_2 \cdot X$
23 If $(e_x, e_y) = (0, 0)$
24    Return 0
25 Return $r = e_x \mod p$
```

Figure 1: $\mathsf{EC}\,[\mathsf{H}_0] = (\mathsf{EC}[\mathsf{H}_0].\mathsf{Gen}, \mathsf{EC}[\mathsf{H}_0].\mathsf{Sign}, \mathsf{EC}[\mathsf{H}_0].\mathsf{Verify})$: ECDSA signature scheme over to elliptic curve $\mathbb{E}$ using hash function $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$.

The RKA from Lemma 3.1 can be extended to an RKA between the schemes $\mathsf{EC}[\mathsf{H}_0]$ and $\mathsf{REC}[\mathsf{H}_1]$ such that a valid signature under pk_0 for a *prefixed* message $\mathsf{pm} \leftarrow (\mathsf{pk}_1, m)$ in $\mathsf{EC}[\mathsf{H}_0]$ is also valid in $\mathsf{REC}[\mathsf{H}_1]$ under pk_1 for message m. This RKA allows to transfer a valid signature from $\mathsf{EC}[\mathsf{H}_0]$ to a valid signature in $\mathsf{REC}[\mathsf{H}_1]$ and vice versa in case pk_0 and pk_1 satisfy the relation from Lemma 3.1. We formally present this RKA in the following Lemma.

Lemma 3.2 *Let* H_0, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions (modeled as random oracles). Let* $m \in \{0,1\}^*$ *and suppose that* $\sigma = (r, s)$ *is a valid signature on message* $\mathsf{pm} \leftarrow (\mathsf{pk}_1, m)$ *w.r.t.* $\mathsf{EC}[\mathsf{H}_0]$ *and public key* pk_0*, i.e.,* $\mathsf{EC}[\mathsf{H}_0].\mathsf{Verify}(\mathsf{pk}_0, \sigma, \mathsf{pm}) = 1$*. Furthermore, suppose that* $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$ *where* $\rho = \frac{H_0(\mathsf{pm}) - H_1(\mathsf{pm})}{r} \pmod p$*. Then* σ *is also a valid signature on message* m *w.r.t.* $\mathsf{REC}[\mathsf{H}_1]$ *and public key* pk_1*, i.e.,* $\mathsf{REC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m) = 1$.

Proof of Lemma 3.2. We have to show that $\mathsf{REC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m) = 1$ for $\mathsf{pk}_1 = \mathsf{pk}_0 + \rho \cdot G$ and $\rho = \frac{H_0(\mathsf{pm}) - H_1(\mathsf{pm})}{r} \pmod p$, where $\mathsf{pm} \leftarrow (\mathsf{pk}_1, m)$. Note that $\sigma = (r, s)$, where $s = t^{-1}(\mathsf{H}_0(\mathsf{pm}) + r\mathsf{sk}_0) \pmod p$ and r represents the x-coordinate of the elliptic curve point $t \cdot G$ for $t \xleftarrow{\$} \mathbb{Z}_p$. As shown in figure 7, $\mathsf{REC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, m)$ first computes the prefixed message $\mathsf{pm} \leftarrow (\mathsf{pk}_1, m)$ and then runs $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}_1, \sigma, \mathsf{pm})$. The rest follows from the proof of Lemma 3.1 with $m_0 = m_1 = \mathsf{pm}$. ∎

3.1 Security analysis of REC

In this section, we analyze the one-per message unforgeability of the honestly rerandomizable signature scheme, or in short the **uf-cma-hrk1** security of the scheme $\mathsf{REC}[\mathsf{H}_1]$. We prove the following theorem.

Theorem 3.3 *Let* $\mathsf{H}_0, \mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions (modeled as random oracles). Let* $\mathcal{A}$ *be an algorithm that plays in the game* $\textbf{uf-cma-hrk1}_{\mathsf{REC}[\mathsf{H}_1]}$*. Then there exists an algorithm* $\mathcal{C}$ *running in roughly the same time as* $\mathcal{A}$*, such that*

$$\mathsf{Adv}^{\mathcal{C}}_{\textbf{uf-cma1}_{\mathsf{EC}[\mathsf{H}_0]}} \geq \left(\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk1}_{\mathsf{REC}[\mathsf{H}_1]}} - \frac{q_{\mathsf{H}_1}^2}{p} \right) \cdot \frac{1}{q}$$

where q_{H_1} *and* q *are the number of random oracle queries and* `Rand` *queries, respectively, that* $\mathcal{A}$ *makes.*

```
Algorithm $\mathsf{REC[H_1].Sign}\,(\mathsf{sk}, m)$
00 $\mathsf{pm} \leftarrow (\mathsf{pk}, m)$
01 $\sigma \leftarrow \mathsf{EC[H_1].Sign}\,(\mathsf{sk}, \mathsf{pm})$
02 Return $\sigma$

Algorithm
$\mathsf{REC[H_1].Verify}\,(\mathsf{pk}, \sigma, m)$
03 $\mathsf{pm} \leftarrow (\mathsf{pk}, m)$
04 Return
$\mathsf{EC[H_1].Verify}\,(\mathsf{pk}, \sigma, \mathsf{pm})$

Algorithm
$\mathsf{REC[H_1].RandSK}\,(\mathsf{sk}; \rho)$
00 $\mathsf{sk}' \leftarrow (\mathsf{sk} + \rho) \mod p$
01 Return $\mathsf{sk}'$

Algorithm
$\mathsf{REC[H_1].RandPK}\,(\mathsf{pk}; \rho)$
02 $\mathsf{pk}' \leftarrow (\mathsf{pk} + \rho \cdot G)$
03 Return $\mathsf{pk}'$
```

Figure 2: Key-prefixed version of the ECDSA signature scheme with perfectly rerandomizable keys $\mathsf{REC[H_1]}$:= ($\mathsf{REC[H_1].Gen} = \mathsf{EC[H_1].Gen}$, $\mathsf{REC[H_1].Sign}$, $\mathsf{REC[H_1].Verify}$, $\mathsf{REC[H_1].RandSK}$, $\mathsf{REC[H_1].RandPK}$) based on the ECDSA signature scheme $\mathsf{EC[H_1]}$. Above $\mathsf{H_1}\colon \{0,1\}^* \to \mathbb{Z}_p$ denotes a hash function.

Due to space limitations, we present the full proof of Theorem 3.3 in Appendix A. We now give some intuition on how we overcome the main difficulties in our simulation. At a high level, the idea is to reduce the **uf-cma-hrk1** security of the additively rerandomizable ECDSA construction $\mathsf{REC[H_1]}$ from the **uf-cma1** security of ECDSA construction $\mathsf{EC[H_0]}$. Therefore, the proof essentially consists of building a reduction $\mathcal{C}$ trying to come up with a valid forgery to win the **uf-cma1**$_{\mathsf{EC[H_0]}}$ game, by simulating the **uf-cma-hrk1**$_{\mathsf{REC[H_1]}}$ game to adversary $\mathcal{A}$ using the RKA from Lemma 3.2. In the **uf-cma1**$_{\mathsf{EC[H_0]}}$ game, $\mathcal{C}$ obtains a public key $\mathsf{pk}_{\mathcal{C}}$ from its challenger. It can query an oracle `Sign` to get signatures w.r.t. $\mathsf{pk}_{\mathcal{C}}$. $\mathcal{C}$ also has access to a random oracle $\mathsf{H_0}$. $\mathcal{C}$'s goal is to somehow embed its public key $\mathsf{pk}_{\mathcal{C}}$ in one of the rerandomized public keys pk^* under which $\mathcal{A}$ eventually returns a forgery $(\mathsf{pk}^*, \sigma^*, m^*)$. The hope is that $\mathcal{C}$ can use $(\mathsf{pk}^*, \sigma^*, m^*)$ to win its own game **uf-cma1**$_{\mathsf{EC[H_0]}}$.

In more detail, $\mathcal{C}$'s strategy works as follows. Instead of directly using $\mathsf{pk}_{\mathcal{C}}$, $\mathcal{C}$ generates the challenge public key for $\mathcal{A}$ by additively shifting $\mathsf{pk}_{\mathcal{C}}$ with a freshly sampled $\tilde{\rho} \xleftarrow{\$} \mathcal{R}$, i.e., $\mathsf{pk} \leftarrow \mathsf{pk}_{\mathcal{C}} - \tilde{\rho} \cdot G$. When $\mathcal{A}$ asks for a signature under a key $\mathsf{pk}' = \mathsf{pk} + \rho \cdot G$, $\mathcal{C}$ can simulate such signatures by querying its `Sign` oracle and employing the RKA from Lemma 3.2. This is because, to the adversary $\mathcal{A}$, pk' looks like a rerandomization of pk, while in fact, it is derived from $\mathsf{pk}_{\mathcal{C}}$ as $\mathsf{pk}' = \mathsf{pk} + \rho \cdot G = (\mathsf{pk}_{\mathcal{C}} - \tilde{\rho} \cdot G) + \rho \cdot G$. To make this simulation work, the random oracle $\mathsf{H_1}$ must be carefully programmed by $\mathcal{C}$ such that the relation between ρ, $\mathsf{H_0}$ and $\mathsf{H_1}$ satisfies $\mathsf{H_1}(m) = \mathsf{H_0}(m) - r \cdot \rho \pmod p$ (according to Lemma 3.2), where $(r, s) := \sigma$ is the signature[3].Note that, due to the programming of the random oracle, the first simulated signature for every message and randomness pair (m, ρ) fully determines $\mathsf{H_1}(m)$. Hence, the simulated signing oracle in **uf-cma-hrk1**$_{\mathsf{REC[H_1]}}$ can be queried at most once on every input pair (m, ρ). $\mathcal{C}$'s strategy to win **uf-cma1**$_{\mathsf{EC[H_0]}}$ is to embed $\tilde{\rho}$ at random as an answer to one of the `Rand` queries in **uf-cma-hrk1**$_{\mathsf{REC[H_1]}}$. For signing queries w.r.t. $\tilde{\mathsf{pk}}$, $\mathcal{C}$ does not reprogram $\mathsf{H_1}$; instead, it uses $\mathsf{H_0}$ and signatures obtained from the signing oracle in **uf-cma1**$_{\mathsf{EC[H_0]}}$ directly. If $\mathcal{A}$ returns a valid forgery σ^* w.r.t. to $\mathsf{pk}^* = \tilde{\mathsf{pk}} = \mathsf{pk} + \tilde{\rho} \cdot G$, then $\mathcal{C}$ can simply use this forgery to win the **uf-cma1**$_{\mathsf{EC[H_0]}}$ game. This is because $\mathsf{pk}^* = \mathsf{pk} + \tilde{\rho} \cdot G = \mathsf{pk}_{\mathcal{C}} - \tilde{\rho} \cdot G + \tilde{\rho} \cdot G = \mathsf{pk}_{\mathcal{C}}$. Note that pk^* is *the only key* for which the forgery σ^* is valid in game **uf-cma1**$_{\mathsf{EC[H_0]}}$. For any other key pk', the simulation of the signing oracle in **uf-cma-hrk1**$_{\mathsf{REC[H_1]}}$ requires to reprogram $\mathsf{H_1}$ on any message that is prefixed with pk'. Since this involves a signing query on that very message to the signing oracle in **uf-cma1**$_{\mathsf{EC[H_0]}}$, the forgery would no longer be fresh in the latter game. This guessing on $\mathcal{C}$'s part is also the reason that our reduction is not tight.

4 A Model for Hierarchical Deterministic Wallets

In this section, we introduce a formal model for hierarchical deterministic wallets. This model closely reflects the BIP32 specification [Wik18] with only minor differences which we list in Section 6. At a high

[3]An important aspect of this simulation is that $\mathcal{C}$ can program $\mathsf{H_1}$ whenever it observes a query m to $\mathsf{H_1}$ that is prefixed with a previously rerandomized key. In particular, this can be done *before* m is ever queried to the signing oracle in **uf-cma-hrk1**$_{\mathsf{REC[H_1]}}$.

level, a hierarchical deterministic wallet scheme can be visualized as a tree, where every node in the tree corresponds to a wallet. As is usual in a tree structure, the scheme originates from a root node, which contains a pair of master keys - a master public key mpk and a master secret key msk as well as a seed $\mathsf{ch}_{0,0}$ which we will refer to as chaincode from now on. We say that the root node is located at level 0 of the tree. The root can create a child node at level 1 and position t by deriving a new key pair $(\mathsf{pk}_{1,t}, \mathsf{sk}_{1,t})$ and a chaincode $\mathsf{ch}_{1,t}$ from its master keys and chaincode $\mathsf{ch}_{0,0}$. This child node represents a new wallet that is initiated with the key pair $(\mathsf{pk}_{1,t}, \mathsf{sk}_{1,t})$ and chaincode $\mathsf{ch}_{1,t}$ and using these values it can in turn create a child node for level 2. This child creation process can continue recursively. Note, however, that a node at level i can only create children for the *immediate* lower level, i.e., for level $i+1$.

In our model, we distinguish between two different kinds of nodes, namely *non-hardened* and *hardened* nodes. Non-hardened nodes are, in essence, the nodes as discussed above, i.e., nodes that can be used for child creation at the next lower level. We assume that the public key and the chaincode of a non-hardened node can be corrupted by an adversary, whereas the secret key remains protected. One might think of non-hardened nodes as wallets in the hot/cold wallet setting, where the hot wallet stores the public key, the cold wallet stores the secret key and the chaincode is provided to both wallets. While the hot wallet is permanently online and thereby vulnerable to attacks, the cold wallet stays offline for the majority of the time and is therefore protected against attacks. To create a non-hardened child node at level i and at position t, its parent must generate the child node's key pair $(\mathsf{pk}_{i,t}, \mathsf{sk}_{i,t})$ and chaincode $\mathsf{ch}_{i,t}$. We model the derivation of these values in such a way that the derivation process of $\mathsf{sk}_{i,t}$ involves the parent's secret key, while the derivation of $\mathsf{pk}_{i,t}$ and $\mathsf{ch}_{i,t}$ requires only the parent's public key and chaincode (i.e., it is independent of the parent's secret key).

Hardened nodes, on the other hand, represent the leaves of the tree, i.e., we do not consider any child derivation from hardened nodes[4]. However, in comparison to non-hardened nodes we allow secret key leakage, along with public key and chaincode leakage for hardened nodes. That is, we consider full corruption of hardened nodes. Our security goal is that the secret key leakage of a hardened node does not affect the security of any other node in the tree. As opposed to non-hardened nodes, the creation process of a hardened child node requires the secret key of the parent node, i.e., even for the derivation of the child's public key and chaincode . The tree structure of a hierarchical deterministic wallet scheme, containing hardened as well as non-hardened nodes can be found in Figure 4.

While hardened nodes clearly exhibit stronger security guarantees than non-hardened nodes, the advantage of non-hardened nodes lies in the child creation process. We will illustrate this advantage in the following example. In a company there might be trusted and untrusted employees. Trusted employees operate a non-hardened node, as they are trusted to properly protect their secret key, e.g., by storing it in a cold wallet. On the other hand, untrusted employees have to operate a hardened node as they might leak their secret key or simply get compromised. Assume a trusted employee maintains a non-hardened node with key pair $(\mathsf{pk}_{i,t}, \mathsf{sk}_{i,t})$ and chaincode $\mathsf{ch}_{i,t}$. Further assume that the node is operated in a hot/cold wallet setting, i.e., the tuple $(\mathsf{sk}_{i,t}, \mathsf{ch}_{i,t})$ is stored in a cold wallet and the tuple $(\mathsf{pk}_{i,t}, \mathsf{ch}_{i,t})$ is stored in a hot wallet. If the employee wishes to receive payments to different public addresses, it can simply generate these addresses by deriving non-hardened child public keys using only the information stored in its hot wallet. In particular, the cold wallet can remain offline during this process. Only when the employee wants to spend the coins it received, it has to use $\mathsf{sk}_{i,t}$ from the cold wallet to generate the secret keys corresponding to the public addresses it generated earlier.

Another example for the usefulness of non-hardened nodes is the following. Consider a company A that operates a non-hardened node with key pair $(\mathsf{pk}_{i,t}, \mathsf{sk}_{i,t})$ and chaincode $\mathsf{ch}_{i,t}$ only to receive payments from a company B. In this case, company A can simply share $\mathsf{pk}_{i,t}$ and $\mathsf{ch}_{i,t}$ with company B, which can then by itself generate non-hardened child public keys and make the payments to those addresses. Note that in this case, company A does not have to be involved in the payment process at all.

Flat Vs Hierarchical Deterministic Wallets. Let us now briefly discuss the main difference between the model for hierarchical deterministic wallets and the setting originally analyzed by Das et. al [DFL19] which we denote as *the flat model*. The key derivation process in the flat model works in the same way as the non-hardened key derivation in the hierarchical model with the difference that the flat model allows to derive keys only directly from the master key pair. Hardened nodes are not considered in the flat model. Therefore, the flat model basically represents a hierarchical wallet structure with non-hardened leaf nodes at level 1 (see Figure 4). Since the flat model allows only for non-hardened key derivation, the essential

[4]We show in Appendix B that child derivation of hardened nodes is possible under certain conditions.

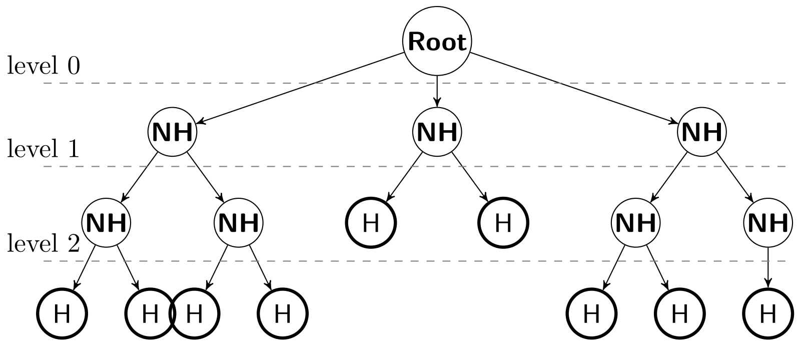


Figure 3: Tree structure of a hierarchical deterministic wallet scheme. Hardened nodes are denoted by H while non-hardened nodes are denoted by NH.

difference to the hierarchical model is that the flat model cannot allow for any secret key leakage as this would render the entire scheme insecure. Hierarchical wallets, on the other hand, introduce hardened nodes whose secret keys can be leaked without affecting the security of any other node in the tree.

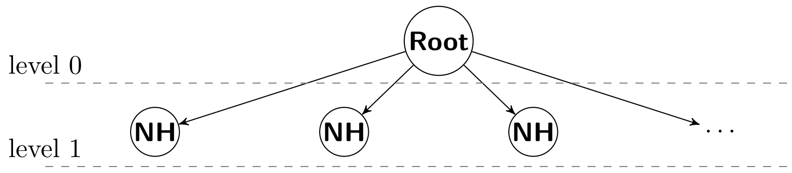


Figure 4: Tree structure of a deterministic wallet scheme in the flat setting.

In the following, we refer to a *tree* as a tuple $(h, n_{0,0}, \mathcal{N}, \mathcal{E})$ if $(\mathcal{N}, \mathcal{E})$ defines a tree of height h with node set $\mathcal{N}$ and edge set $\mathcal{E}$, and a root node $n_{0,0} \in \mathcal{N}$. We denote a directed path p_i^t of length i from the root to a node $n_{i,t} \in \mathcal{N}$ at level i and position t in the tree as the corresponding ordered sequence of edges $p_i^t = (\mathsf{e}_1, \cdots, \mathsf{e}_i) \in \mathcal{E}^i$. A path of length 1 from a node $n_{i-1,s} \in \mathcal{N}$ to a node $n_{i,t} \in \mathcal{N}$ consists of only one edge which we denote as $\mathsf{e}_i^{s,t} \in \mathcal{E}$.

Definition 4.1 (Address Structure). Let $\mathcal{T} = (h, n_{0,0}, \mathcal{N}, \mathcal{E})$ be a tree. Define a labeling of the nodes in $\mathcal{N}$ as follows.

- The root node $n_{0,0}$ is labeled by an address $\mathbf{addr}_{0,0}$.
- For $1 \leq t < |\mathcal{N}|$ and $0 \leq i \leq h$, a node $n_{i,t} \in \mathcal{N}$ is labeled by an address $\mathbf{addr}_{i,t} := (\mathbf{addr}_{0,0}, p_i^t)$.

A tuple $(\mathcal{T}, \mathbf{Addr})$ is said to be an *address structure (with respect to* $\mathcal{T}$*)* if $\mathbf{Addr}$ consists of a set of labels for the nodes in $\mathcal{N}$ that meets the above requirements. A prefix address $\mathbf{addr}_{i,t}^j$ for a node $n_{i,t} \in \mathcal{N}$ with $0 \leq j < i \leq h$ and $t < |N|$ is a vector of length $j+1$ consisting of the first $j+1$ components of $\mathbf{addr}_{i,t} \in \mathbf{Addr}$.

We are now ready to define hierarchical deterministic wallets. In short, these schemes consist of a Setup algorithm, which initializes the root node, hardened and non-hardened secret and public key derivation algorithms $\mathsf{SKDer_H}, \mathsf{PKDer_H}$ and $\mathsf{SKDer_{NH}}, \mathsf{PKDer_{NH}}$ and finally signing and signature verification algorithms Sign and and Verify. We assume that public parameters par are known to all parties and we define appropriate secret and public key sets $\mathcal{SK}$ and $\mathcal{PK}$ respectively. We assume there exists a function $\mathsf{ToPubKey} : \mathcal{SK} \to \mathcal{PK}$ that on input a secret key from $\mathcal{SK}$ outputs the corresponding public key in $\mathcal{PK}$. Formally we have:

Definition 4.2 (Hierarchical Deterministic Wallets). Let $\mathcal{T} = (h, n_{0,0}, \mathcal{N}, \mathcal{E})$ be a tree. A *hierarchical deterministic wallet* scheme is defined w.r.t. an address structure $(\mathcal{T}, \mathbf{Addr})$ and consists of seven algorithms $\mathsf{HDWal} = (\mathsf{Setup}, \mathsf{SKDer_H}, \mathsf{SKDer_{NH}}, \mathsf{PKDer_H}, \mathsf{PKDer_{NH}}, \mathsf{Sign}, \mathsf{Verify})$ which are defined as follows:

- $\mathsf{Setup}(1^\kappa)$: The probabilistic *setup* algorithm takes as input a security parameter 1^κ and outputs a non-hardened master key pair $(\mathsf{msk}_{0,0}, \mathsf{mpk}_{0,0})$ with $\mathsf{msk}_{0,0} \in \mathcal{SK}$, $\mathsf{mpk}_{0,0} \in \mathcal{PK}$ and a chaincode $\mathsf{ch}_{0,0}$.

- $\mathsf{SKDer_H}(\mathsf{sk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$: The deterministic *hardened secret key derivation* algorithm takes as input a secret key $\mathsf{sk}_{i,s} \in \mathcal{SK}$, a chaincode $\mathsf{ch}_{i,s}$, an address $\mathbf{addr}_{i,s} \in \mathbf{Addr}$ for level $i < h$, positions s, t, as well as an edge $\mathsf{e}_{i+1}^{s,t} \in \mathcal{E}$. It outputs a secret key $\mathsf{sk}_{i+1,t} \in \mathcal{SK}$, a chaincode $\mathsf{ch}_{i+1,t}$ and an address $\mathbf{addr}_{i+1,t} \in \mathbf{Addr}$ for level $i+1$ and position t.

- $\mathsf{SKDer_{NH}}(\mathsf{sk}_{i,s}, \mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$: The deterministic *non-hardened secret key derivation* algorithm takes as input a secret key $\mathsf{sk}_{i,s} \in \mathcal{SK}$, a public key $\mathsf{pk}_{i,s} \in \mathcal{PK}$, a chaincode $\mathsf{ch}_{i,s}$, an address $\mathbf{addr}_{i,s} \in \mathbf{Addr}$ for level $i < h$, positions s, t, as well as an edge $\mathsf{e}_{i+1}^{s,t} \in \mathcal{E}$. It outputs a secret key $\mathsf{sk}_{i+1,t} \in \mathcal{SK}$, a chaincode $\mathsf{ch}_{i+1,t}$ and an address $\mathbf{addr}_{i+1,t} \in \mathbf{Addr}$ for level $i+1$ and position t.

- $\mathsf{PKDer_H}(\mathsf{sk}_{i,s}, \mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$: The deterministic *hardened public key derivation* algorithm takes as input a secret key $\mathsf{sk}_{i,s} \in \mathcal{SK}$, a public key $\mathsf{pk}_{i,s} \in \mathcal{PK}$, a chaincode $\mathsf{ch}_{i,s}$, an address $\mathbf{addr}_{i,s} \in \mathbf{Addr}$ for level $i < h$, positions s, t, as well as an edge $\mathsf{e}_{i+1}^{s,t} \in \mathcal{E}$. It outputs a public key $\mathsf{pk}_{i+1,t} \in \mathcal{PK}$, a chaincode $\mathsf{ch}_{i+1,t}$ and an address $\mathbf{addr}_{i+1,t} \in \mathbf{Addr}$ for level $i+1$ and position t.

- $\mathsf{PKDer_{NH}}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$: The deterministic *non-hardened public key derivation* algorithm takes as input a public key $\mathsf{pk}_{i,s} \in \mathcal{PK}$, a chaincode $\mathsf{ch}_{i,s}$, an address $\mathbf{addr}_{i,s} \in \mathbf{Addr}$ for level $i < h$, positions s, t, as well as an edge $\mathsf{e}_{i+1}^{s,t} \in \mathcal{E}$. It outputs a public key $\mathsf{pk}_{i+1,t} \in \mathcal{PK}$, a chaincode $\mathsf{ch}_{i+1,t}$ and an address $\mathbf{addr}_{i+1,t} \in \mathbf{Addr}$ for level $i+1$ and position t.

- $\mathsf{Sign}(\mathsf{sk}_{i,s}, m)$: The probabilistic *signing* algorithm takes as input a secret key $\mathsf{sk}_{i,s}$ and a message m. It outputs a signature σ.

- $\mathsf{Verify}(\mathsf{pk}_{i,s}, m, \sigma)$: The probabilistic *verification* algorithm takes as input a public key $\mathsf{pk}_{i,s}$, a message m and a signature σ. It outputs 0 or 1.

A hierarchical deterministic wallet is *correct*, if a secret and public key pair is derived correctly using the algorithms $\mathsf{SKDer_H}, \mathsf{PKDer_H}$ or $\mathsf{SKDer_{NH}}, \mathsf{PKDer_{NH}}$, the keys represent a valid signing key pair.

We denote keys with subscript nh (e.g., $\mathsf{sk}_{\mathsf{nh},\cdot,\cdot}$ or $\mathsf{pk}_{\mathsf{nh},\cdot,\cdot}$) as *non-hardened* keys and keys with subscript h (e.g., $\mathsf{sk}_{\mathsf{h},\cdot,\cdot}$ or $\mathsf{pk}_{\mathsf{h},\cdot,\cdot}$) as *hardened* keys. A key without the subscript nh or h indicates that it can be both a non-hardened or hardened key.

Definition 4.3 (Correctness of Hierarchical Deterministic Wallets). Let HDWal be a hierarchical deterministic wallet scheme with respect to an address structure $(\mathcal{T}, \mathbf{Addr})$. For any $\mathsf{e}_1^{0,s} \in \mathcal{E}$ and any $(\mathsf{ch}_{0,0}, \mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{mpk}_{\mathsf{nh},0,0}) \in \mathsf{Setup}(1^\kappa)$, we define tuples $(\mathsf{sk}_{\mathsf{h},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s})$ and $(\mathsf{pk}_{\mathsf{h},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s})$ as

$$(\mathsf{sk}_{\mathsf{h},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s}) := \mathsf{SKDer_H}(\mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{ch}_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}_1^{0,s})$$
$$(\mathsf{pk}_{\mathsf{h},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s}) := \mathsf{PKDer_H}(\mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{ch}_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}_1^{0,s})$$

and tuples $(\mathsf{sk}_{\mathsf{nh},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s})$ and $(\mathsf{pk}_{\mathsf{nh}1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s})$ as

$$(\mathsf{sk}_{\mathsf{nh},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s}) := \mathsf{SKDer_{NH}}(\mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{ch}_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}_1^{0,s})$$
$$(\mathsf{pk}_{\mathsf{nh},1,s}, \mathsf{ch}_{1,s}, \mathbf{addr}_{1,s}) := \mathsf{PKDer_{NH}}(\mathsf{mpk}_{\mathsf{nh},0,0}, \mathsf{ch}_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}_1^{0,s}).$$

Further, for any $\mathbf{addr}_{i-1,s} \in \mathbf{Addr}$, and any edge $\mathsf{e}_i^{s,t} \in \mathcal{E}$ we define the tuples $(\mathsf{sk}_{\mathsf{h},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t})$ and $\big(\mathsf{pk}_{\mathsf{h},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}\big)$ recursively as

$$(\mathsf{sk}_{\mathsf{h},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}) := \mathsf{SKDer_H}(\mathsf{sk}_{\mathsf{nh},i-1,s}, \mathsf{ch}_{i-1,s}, \mathbf{addr}_{i-1,s}, \mathsf{e}_i^{s,t})$$
$$\big(\mathsf{pk}_{\mathsf{h},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}\big) := \mathsf{PKDer_H}(\mathsf{sk}_{\mathsf{nh},i-1,s}, \mathsf{ch}_{i-1,s}, \mathbf{addr}_{i-1,t}, \mathsf{e}_i^{s,t})$$

and tuples $(\mathsf{sk}_{\mathsf{nh},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t})$ and $\big(\mathsf{pk}_{\mathsf{nh},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}\big)$ as

$$(\mathsf{sk}_{\mathsf{nh},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}) := \mathsf{SKDer}_{\mathsf{NH}}(\mathsf{sk}_{\mathsf{nh},i-1,s}, \mathsf{ch}_{i-1,s}, \mathbf{addr}_{i-1,s}, \mathsf{e}_i^{s,t})$$
$$(\mathsf{pk}_{\mathsf{nh},i,t}, \mathsf{ch}_{i,t}, \mathbf{addr}_{i,t}) := \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{pk}_{\mathsf{nh},i-1,s}, \mathsf{ch}_{i-1,s}, \mathbf{addr}_{i-1,s}, \mathsf{e}_i^{s,t})$$

HDWal is *correct* if for all $m \in \{0,1\}^*$, all $1 \leq i \leq h$, all $1 \leq t \leq (1 - d^{h+1})/(1 - d)$, and all $(\mathsf{ch}_{0,0}, \mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{mpk}_{\mathsf{nh},0,0}) \in \mathsf{Setup}(1^\kappa)$ it holds that

$$\Pr_{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{h},i,t}, m)}[\mathsf{Verify}(\mathsf{pk}_{\mathsf{h},i,t}, \sigma, m) = 1] = 1$$
$$\wedge \Pr_{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{nh},i,t}, m)}[\mathsf{Verify}(\mathsf{pk}_{\mathsf{nh},i,t}, \sigma, m) = 1] = 1.$$

4.1 Oracles

Let us now describe the general capability and influence that the adversary has over the hierarchical wallet schemes. An adversary is allowed to create new hardened and non-hardened nodes in the tree. Furthermore, the adversary can corrupt the hot wallet of all non-hardened nodes, thereby learning the public key and the chaincode of these nodes, as well as learning the secret key and chaincode of the hardened nodes. As we mentioned earlier, since hardened keys are given to untrustworthy nodes, the adversary is able to corrupt both their hot and cold wallets and as such, we do not consider the hardened nodes to derive new children. One way to look at hardened nodes, is that such nodes are the root of a new tree. We will later show in App. B that an adversary cannot distinguish hardened key pairs from freshly generated keys except with negligible probability. Therefore, our model can be recursively extended to consider settings where the hardened nodes can also derive new keys. Finally, the adversary can query any node on a freely chosen message m and receive a signature for this message. To model the above mentioned capabilities, we describe the oracles which the adversary gets access to in the unlinkability game $\mathbf{unl}_{\mathsf{HDWal}}$ and the unforgeability game $\mathbf{wufcma1}_{\mathsf{HDWal}}$.

Initially, two lists $\mathsf{SK} = \emptyset$ and $\mathsf{CH} = \emptyset$ are initialized. These are used throughout the oracles to bookkeep which secret keys and chaincodes have been leaked to the adversary. In the following, we consider a fixed address structure $(\mathcal{T}, \mathbf{Addr})$.

- **Hardened Child Creation** `HChildO`**:** On inputs an address $\mathbf{addr}_{i,s}$ and an edge $\mathsf{e}_{i+1}^{s,t}$ from $\mathcal{A}$, return $\perp$ if the address $\mathbf{addr}_{i,s}$ belongs to a hardened node or the address $\mathbf{addr}_{i,s}$ is not valid (i.e., $\mathbf{addr}_{i,s} \notin \mathbf{Addr}$). Further, return $\perp$, if the address $\mathbf{addr}_{i+1,t}$ exists already. Otherwise, compute the keys and chaincode $(\mathsf{sk}_{\mathsf{h},i,s}, \mathsf{pk}_{\mathsf{h},i,s})$ and $\mathsf{ch}_{i,s}$ for the node $\mathbf{addr}_{i,s}$ by recursively deriving keys along the path in the tree, starting from the first node in the path that has already been assigned a key. Create a hardened child with address $\mathbf{addr}_{i+1,t}$ as follows. Generate keypair $(\mathsf{sk}_{\mathsf{h},i+1,t}, \mathsf{pk}_{\mathsf{h},i+1,t})$ by executing both secret and public key derivation algorithms.

$$(\mathsf{sk}_{\mathsf{h},i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t}) \leftarrow \mathsf{SKDer}_{\mathsf{H}}(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$$
$$(\mathsf{pk}_{\mathsf{h},i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t}) \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t}).$$

  Return $\mathsf{pk}_{\mathsf{h},i+1,t}$.

- **Non-Hardened Child Creation** `NHChildO`**:** On inputs an address $\mathbf{addr}_{i,s}$ and an edge $\mathsf{e}_{i+1}^{s,t}$ from $\mathcal{A}$, return $\perp$ if the address $\mathbf{addr}_{i,s}$ belongs to a hardened node or the address $\mathbf{addr}_{i,s}$ is not valid (i.e., $\mathbf{addr}_{i,s} \notin \mathbf{Addr}$). Further, return $\perp$, if the address $\mathbf{addr}_{i+1,t}$ exists already. Otherwise, compute the keys and chaincode $(\mathsf{sk}_{\mathsf{h},i,s}, \mathsf{pk}_{\mathsf{h},i,s})$ and $\mathsf{ch}_{i,s}$ for the node $\mathbf{addr}_{i,s}$ by recursively deriving keys along the path in the tree, starting from the first node in the path that has already been assigned a key. Create a non-hardened child with address $\mathbf{addr}_{i+1,t}$ as follows. Generate keypair $(\mathsf{sk}_{\mathsf{nh},i+1,t}, \mathsf{pk}_{\mathsf{nh},i+1,t})$ by executing both key derivation algorithms

$$(\mathsf{sk}_{\mathsf{nh},i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t}) \leftarrow \mathsf{SKDer}_{\mathsf{NH}}(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$$
$$(\mathsf{pk}_{\mathsf{nh},i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t}) \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{pk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t}).$$

  Return $\mathsf{pk}_{\mathsf{nh},i+1,t}$.

- **Signing** `HDSignO`**:** On input message m and an address $\mathbf{addr}_{i,s}$ from $\mathcal{A}$, proceed as follows. Return $\perp$ if the address $\mathbf{addr}_{i,s}$ is not valid (i.e., $\mathbf{addr}_{i,s} \notin \mathbf{Addr}$). Further, check if $\mathbf{addr}_{i,s}$ has already been queried to either `NHChildO` or `HChildO` and return $\perp$ if this is not the case. Let $\mathsf{sk}_{i,s}$ be the secret key for the node with address $\mathbf{addr}_{i,s}$. Then compute a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{i,s}, m)$, add m to the message list $\mathsf{SigList}[\mathbf{addr}_{i,s}]$ and return σ.[5]

- **Chaincode Leakage** `CHLeakO`**:** On input an address $\mathbf{addr}_{i,s}$ from $\mathcal{A}$, check if $\mathbf{addr}_{i,s}$ has already been queried to either `NHChildO` or `HChildO` and return $\perp$ if this is not the case. Set $\mathsf{CH}[\mathbf{addr}_{i,s}] = 1$ to denote that the chaincode $\mathsf{ch}_{i,s}$ of address $\mathbf{addr}_{i,s}$ has been leaked and return $(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s})$.

- **Secret Key Leakage (for hardened node)** `SKLeakO`**:** On input an address $\mathbf{addr}_{i,s}$ from $\mathcal{A}$, check if the address is that of the root, i.e., $\mathbf{addr}_{i,s} = \mathbf{addr}_{0,0}$ or if the address belongs to a non-hardened node; in this case, return $\perp$. Further, check if $\mathbf{addr}_{i,s}$ has already been queried to either `NHChildO` or `HChildO` and return $\perp$ if this is not the case. Else, set $\mathsf{SK}[\mathbf{addr}_{i,s}] = 1$ and $\mathsf{CH}[\mathbf{addr}_{i,s}] = 1$ to denote that the secret key $\mathsf{sk}_{h,i,s}$ and the chaincode $\mathsf{ch}_{i,s}$ of address $\mathbf{addr}_{i,s}$ have been leaked and return $(\mathsf{sk}_{h,i,s}, \mathsf{ch}_{i,s})$.

4.2 Unlinkability

Intuitively, the notion of unlinkability for hierarchical deterministic wallets guarantees that public keys in the tree, i.e., public keys that have been derived directly or indirectly from the master key of the tree root, cannot be distinguished from from a freshly generated public key. More concretely, the distribution of public keys from the tree should be computationally indistinguishable from a distribution of public keys that have been derived from an independently chosen master key. While this is a valuable privacy notion, it does not quite model practical scenarios in the hot/cold wallet setting. Recall that this setting assumes public keys and chaincodes to be stored in hot wallets, which are prone to corruptions. Therefore, we extend the unlinkability notion as described above in the following way. We consider hot wallet corruption upon which the public key and chaincode of the corrupted wallet are leaked. This extended notion gives more power to the adversary and is more close to the capabilities that an adversary has in real life scenarios. Naturally, the adversary can distinguish the distribution of keys derived from public keys of corrupted hot wallets from a distribution of public keys that have been derived from an independently chosen master key. Therefore, in our new unlinkability notion the adversary should not be able to distinguish the distribution of keys derived from non-compromised hot wallets and keys derived from independently chosen master keys.

In the following we describe the unlinkability game $\mathbf{unl}_{\mathsf{HDWal}}$ with respect to a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In the first step of the game, the challenger generates a fresh master key pair and a chaincode via the execution of $\mathsf{Setup}(1^\kappa)$. The adversary receives the master public key as input and obtains access to all oracles as described in subsection 4.1. At some point, the adversary outputs an address $\mathbf{addr}_{i,s}$ and an edge $\mathsf{e}_{i+1}^{s,t}$ and receives a public key from the challenger. This public key is either the correct key for the node at address $\mathbf{addr}_{i,s}$ or a public key derived for a random address from a fresh master public key. $\mathcal{A}$ wins the game if it can successfully distinguish these two scenarios. In the following we give a detailed description of the game $\mathbf{unl}_{\mathsf{HDWal}}$:

Game $\mathbf{unl}_{\mathsf{HDWal}}$:

- **Setup Phase:** The challenger computes $(\mathsf{ch}_{0,0}, \mathsf{msk}_{0,0}, \mathsf{mpk}_{0,0}) \leftarrow \mathsf{Setup}(1^\kappa)$ and sends $\mathsf{mpk}_{0,0}$ to $\mathcal{A}$.

- **Online Phase:** On input the security parameter and the master public key $\mathsf{mpk}_{0,0}$, the adversary $\mathcal{A}$ is allowed to make queries to the oracles as explained in subsection 4.1.

- **Output Phase:** Eventually, $\mathcal{A}$ chooses an address $\mathbf{addr}_{i,s}$, an edge $\mathsf{e}_{i+1}^{s,t}$ and a value $c \in \{\mathsf{h}, \mathsf{nh}\}$ and sends them to the challenger. Let $(\mathsf{sk}_{i,s}, \mathsf{pk}_{i,s})$ be the key pair and $\mathsf{ch}_{i,s}$ the chaincode of the node at address $\mathbf{addr}_{i,s}$. If the address $\mathbf{addr}_{i,s}$ belongs to a hardened node, $\mathcal{C}$ returns $\perp$. Otherwise, the challenger chooses a bit $b \xleftarrow{\$} \{0,1\}$ and generates a public key $\mathsf{pk}_{i+1,t}$ as follows:

  - If $b = 0$:
    - $\underline{\text{If } c = \mathsf{h}\text{:}}$ $\mathcal{C}$ computes $(\mathsf{pk}_{\mathsf{h},i+1,t}, \cdot, \cdot) \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$.

[5]In case of one-per message unforgeability, the oracle aborts if it has been queried previously on input $(m, \mathbf{addr}_{i,s})$.

    * If $c = \mathsf{nh}$: If the chaincode for $\mathbf{addr}_{i,s}$ or any of its prefix addresses has been leaked, i.e., $\mathsf{CH}[\mathbf{addr}^j_{i,s}] = 1$, for any $j < i$, then $\mathcal{C}$ returns $\perp$. Else, $\mathcal{C}$ computes $(\mathsf{pk}_{\mathsf{nh},i+1,t}, \cdot, \cdot) \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{pk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}^{s,t}_{i+1})$.

  - If $b = 1$: The challenger computes $(\mathsf{ch}'_{0,0}, \mathsf{msk}'_{0,0}, \mathsf{mpk}'_{0,0}) \leftarrow \mathsf{Setup}(1^\kappa)$.
    * If $c = \mathsf{h}$: $\mathcal{C}$ derives a public key $\mathsf{pk}'_{\mathsf{h},1,t} \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{msk}'_{0,0}, \mathsf{ch}'_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}^{0,t}_1)$.
    * If $c = \mathsf{nh}$: If the chaincode for $\mathbf{addr}_{i,s}$ or any of its prefix addresses has been leaked, i.e., $\mathsf{CH}[\mathbf{addr}^j_{i,s}] = 1$, for any $j < i$, then $\mathcal{C}$ returns $\perp$. Else $\mathcal{C}$ derives a public key $\mathsf{pk}'_{\mathsf{nh},1,t} \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{mpk}'_{0,0}, \mathsf{ch}'_{0,0}, \mathbf{addr}_{0,0}, \mathsf{e}^{0,t}_1)$.
  - Based on the value of b and c, the challenger sends to the adversary either $\mathsf{pk}_{\mathsf{h},i+1,t}$ or $\mathsf{pk}_{\mathsf{nh},i+1,t}$ or $\mathsf{pk}'_{\mathsf{h},1,t}$ or $\mathsf{pk}'_{\mathsf{nh},1,t}$.

- The adversary can continue to make oracle queries under the restrictions as mentioned above.
- Eventually, $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.

We define the advantage of an adversary $\mathcal{A}$ in $\mathbf{unl}_{\mathsf{HDWal}}$ as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathbf{unl}_{\mathsf{HDWal}}} := \left| \Pr[\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal}} = 1] - \frac{1}{2} \right| .$$

On Forward Unlinkability The model of hierarchical wallets as defined in Definition 4.2 in Section 4 is stateless. In other words, each node in the tree maintains a fixed chaincode $\mathsf{ch}_{i,s}$ which is used as an input parameter for the child key derivation algorithms. If the (non-hardened) public key $\mathsf{pk}_{i,s}$ as well as the chaincode $\mathsf{ch}_{i,s}$ of a node are leaked (e.g., due to a hot wallet corruption of the node in the hot/cold wallet setting), then the adversary can as well compute the non-hardened keys in the entire sub-tree under $\mathsf{pk}_{i,s}$. Consequently, unlinkability of the sub-tree is lost. To enhance the unlinkability property, we can extend our model to a stateful variant where, each node maintains a state $\mathsf{St}^t_{i,s}$. On every child key derivation, the state of the node is refreshed to a new state $\mathsf{St}^{t+1}_{i,s}$. As a result of this modification, we can guarantee *forward unlinkability* for hierarchical wallets, which is similar to the standard notion of *forward security.* Precisely, on a hot wallet corruption, the adversary learns the *current state* $\mathsf{St}^t_{i,s}$ and the public key $\mathsf{pk}_{i,s}$ of a node. However, the existing children of this node were derived from earlier states $\mathsf{St}^{t'}_{i,s}$, for $t' < t$ - which are not known to the adversary. Thus it can no longer break the unlinkability of the existing child keys in the sub-tree under $\mathsf{pk}_{i,s}$. However, it would be able to link any future child keys derived from $\mathsf{pk}_{i,s}$.

4.3 Unforgeability

The notion of unforgeability for hierarchical deterministic wallets in the hot/cold wallet setting guarantees that an adversary cannot forge a signature of any uncorrupted node in the tree. In our model, non-hardened keys are always stored in hot/cold wallets, i.e., the secret keys are secured in the cold wallet storage, which cannot be corrupted by an adversary. Hardened keys, on the other hand, can be stored on any device and are thereby prone to corruption. Therefore, we allow an adversary to corrupt hardened secret keys, while non-hardened secret keys must remain uncorrupted.

In more detail, the unforgeability game proceeds as follows. The challenger generates a master key pair and a chaincode via the execution of $\mathsf{Setup}(1^\kappa)$. The adversary receives the master public key and obtains access to the oracles as described in subsection 4.1. Eventually, the adversary outputs a forgery, i.e., a message and a signature for a specific node in the tree. The adversary wins the game, if the signature is valid, the message has not been queried to the signing oracle `HDSignO` for this specific node before and the cold wallet of the node is uncorrupted. We note that a slightly weaker variant of unforgeability for hierarchical deterministic wallets is the notion of *one-per message unforgeability*, where the security game proceeds exactly as the game of the unforgeability notion with the difference that the adversary is allowed to query the `HDSignO` oracle only once for each message/address pair. We now give a detailed description of the unforgeability game $\mathbf{wufcma1}_{\mathsf{HDWal}}$.

Game $\mathbf{wufcma1}_{\mathsf{HDWal}}$:

- **Setup Phase:** The challenger computes $(\mathsf{ch}_{0,0}, \mathsf{msk}_{0,0}, \mathsf{mpk}_{0,0}) \leftarrow \mathsf{Setup}(1^n)$ and sends $\mathsf{ch}_{0,0}$ and $\mathsf{mpk}_{0,0}$ to $\mathcal{A}$.
- **Online Phase:** On input the security parameter, the adversary $\mathcal{A}$ is allowed to make queries to the oracles as explained in subsection 4.1.
- **Output Phase:** Eventually, $\mathcal{A}$ outputs a public key pk_{i^*,s^*}, a message m^*, an address $\mathbf{addr}_{i^*,s^*}$ and a signature σ^*. $\mathcal{A}$ wins if all of the following conditions hold,
  - $\mathsf{Verify}(pk_{i^*,s^*}, \sigma^*, m^*) = 1$
  - $m^* \notin \mathsf{SigList}[\mathbf{addr}_{i^*,s^*}]$
  - Either $\mathbf{addr}_{i^*,s^*}$ belongs to a non-hardened node or $\mathbf{addr}_{i^*,s^*}$ belongs to a hardened node and its secret key has not been corrupted, i.e., $\mathsf{SK}[\mathbf{addr}_{i^*,s^*}] = 0$.

We define the advantage of an adversary $\mathcal{A}$ in $\mathbf{wufcma1}_{\mathsf{HDWal}}$ as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathbf{unl}_{\mathsf{HDWal}}} := \Pr[\mathbf{wufcma1}^{\mathcal{A}}_{\mathsf{HDWal}} = 1].$$

5 Generic Construction

In this section, we first show how to generically construct a hierarchical deterministic wallet scheme HDWal from a signature scheme with perfectly rerandomizable keys RSig = (RSig.Gen, RSig.RandSK, RSig.RandPK, RSig.Sign, RSig.Verify). We denote the construction of HDWal with respect to a signature scheme with rerandomizable keys RSig by HDWal[RSig]. Our generic construction HDWal[RSig] uses internally a hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{R} \times \{0,1\}^\kappa$. We detail our construction in Figure 5. Subsequently, we analyze the security of our generic construction by proving the unlinkability and the unforgeability properties of HDWal[RSig]. Due to space limitations, we present the full proofs for unlinkability and unforgeability of HDWal[RSig] in Appendices B and C. In the following subsection, we present the theorem that states that HDWal[RSig] satisfies $\mathbf{wufcma1}_{\mathsf{HDWal}}$ security with a loss in the security reduction. We then show that this loss is indeed unavoidable which means that our security reduction is optimal.

5.1 Unforgeability of Generic Construction

We now analyze the unforgeability property of our generic construction HDWal[RSig] of a hierarchical wallet. We require the following properties from the underlying signature scheme RSig. RSig must satisfy (1) the definition of a signature scheme with rerandomizable keys as well as (2) a transitive property of the keys. We formally define the latter below.

Definition 5.1 (Transitive Rerandomization). Let RSig = (RSig.Gen, RSig.Sign, RSig.Verify,RSig.RandSK, RSig.RandPK) be a signature scheme with perfectly rerandomizable keys. We say that RSig *transitively rerandomizes* if there exists an operation $\odot : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ s.t. for all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{RSig.Gen}(\mathsf{par})$ and all $(\rho, \rho') \in \mathcal{R} \times \mathcal{R}$, the values $(\mathsf{sk}', \mathsf{pk}'), (\mathsf{sk}'', \mathsf{pk}''), \tilde{\rho}$ which are defined as

$$\begin{aligned}
&(\mathsf{sk}', \mathsf{pk}') \leftarrow (\mathsf{RSig.RandSK}(\mathsf{sk}; \rho), \mathsf{RSig.RandPK}(\mathsf{pk}; \rho))\\
&(\mathsf{sk}'', \mathsf{pk}'') \leftarrow (\mathsf{RSig.RandSK}(\mathsf{sk}'; \rho'), \mathsf{RSig.RandPK}(\mathsf{pk}'; \rho')),\\
&\tilde{\rho} = \rho \odot \rho' \text{ satisfy}\\
&(\mathsf{sk}'', \mathsf{pk}'') = (\mathsf{RSig.RandSK}(\mathsf{sk}; \tilde{\rho}), \mathsf{RSig.RandPK}(\mathsf{pk}; \tilde{\rho})).
\end{aligned}$$

Definition 5.2 (Invertible Rerandomization). Let RSig = (RSig.Gen, RSig.Sign, RSig.Verify,RSig.RandSK, RSig.RandPK) be a signature scheme with perfectly rerandomizable keys. We say that RSig has *invertible rerandomization* if there exist (efficient) algorithms RandSK^{-1} and RandPK^{-1} s.t. for all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{RSig.Gen}(\mathsf{par})$ and all $\rho \in \mathcal{R}$ it holds

$$\begin{aligned}
\mathsf{sk} &= \mathsf{RandSK}^{-1}(\mathsf{RSig.RandSK}(\mathsf{sk}; \rho); \rho)\\
\mathsf{pk} &= \mathsf{RandPK}^{-1}(\mathsf{RSig.RandPK}(\mathsf{pk}; \rho); \rho)
\end{aligned}$$

We note that the signature schemes with rerandomizable keys based on Schnorr [FKM+16], BLS [DFL19] and ECDSA (additive variant presented in Section 3 of this work and multiplicative variant presented in [DFL19]) all satisfy the properties of transitive rerandomization and invertible rerandmization as defined in Definitions 5.1, 5.2. For the Schnorr, BLS and additive ECDSA based schemes, the $\odot$ operation is a simple addition, while for the multiplicative ECDSA scheme it is a multiplication (modulo the group order p). Below we state our theorem for the one-per message unforgeability property of $\mathsf{HDWal}[\mathsf{RSig}]$.

Theorem 5.3 *Let* $\mathsf{HDWal}[\mathsf{RSig}]$ *be the construction defined in Figure 5, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{R} \times \{0,1\}^\kappa$ *be a hash function modeled as a random oracle and let* RSig *be a signature scheme with rerandomizable keys that satisfies the property of transitive rerandomization and invertible rerandomization as in Definitions 5.1, 5.2. Let* $\mathcal{A}$ *be an adversary playing in the game* $\mathbf{wufcma1}^{\mathcal{A}}_{\mathsf{HDWal}[\mathsf{RSig}]}$*, then there exists an algorithm* $\mathcal{C}$ *running in roughly the same time as* $\mathcal{A}$*, and that makes as many queries to the oracle* `Rand` *in* **uf-cma-hrk1** *as* $\mathcal{A}$ *makes queries to* `NHChildO`/`HChildO` *such that*

$$\mathsf{Adv}^{\mathcal{C}}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} \geq \frac{1}{4e(q_{sk}+1)} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal}[\mathsf{RSig}]}}.$$

where q_{sk} *is the number of* `SKLeakO` *oracle queries from* $\mathcal{A}$*.*

We stated Theorem 5.3 w.r.t. the one-per message unforgeability notions of hierarchical deterministic wallet schemes and signature schemes with reradomizable keys, because these notions are sufficient in the setting of deterministic wallets. This is because wallets sign each unique transaction at most once. However, we note that we can likewise state and prove the above theorem with respect to the standard unforgeability notions, i.e., the notions that do not restrict the adversary to obtain at most one signature on a specific message.

We provide the full proof of Theorem 5.3 in Appendix C. Essentially, the proof exhibits an adversary $\mathcal{C}$ who uses the adversary $\mathcal{A}$ who plays in game $\mathbf{wufcma1}^{\mathcal{A}}_{\mathsf{HDWal}[\mathsf{RSig}]}$ to win its own game $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{C}}_{\mathsf{RSig}}$. The main idea of our proof is that $\mathcal{C}$ guesses in advance which hardened nodes $\mathcal{A}$ might corrupt (i.e., calls the `SKLeakO` oracle on). In case the guess of $\mathcal{C}$ is wrong, $\mathcal{C}$ cannot answer all `SKLeakO` oracle queries from $\mathcal{A}$ and therefore has to abort. This leads to a polynomial loss in the number of `SKLeakO` oracle queries (i.e., q_{sk}) in $\mathcal{C}$'s advantage in its $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{C}}_{\mathsf{RSig}}$ game. We use Coron's technique as presented in [Cor02] to bound this loss.

Interestingly, the following theorem states that this loss in the advantage of $\mathcal{C}$ is inherent and that, in fact, there does not exist a tighter security reduction. In Appendix D, we recall the security notion of *unforgeability under rerandomized keys* $\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}$ for a signature scheme with rerandomizable keys RSig as introduced in [FKM+16] and prove Theorem 5.4. Below, we denote as $\mathcal{A}_1^{\mathcal{A}_2}$ that $\mathcal{A}_1$ has black-box access to $\mathcal{A}_2$. In particular, it does not rewind $\mathcal{A}_2$.

Theorem 5.4 *Let* HDWal *be an algorithm such that for any signature scheme with rerandomizable keys* RSig*,* $\mathsf{HDWal}^{\mathsf{RSig}}$ *is a hierarchical deterministic wallet scheme. Moreover, suppose that there is a reduction* $\mathcal{R}$ *such that for every signature scheme with rerandomizable keys* RSig *and every adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}}$ *with* $\epsilon_{\mathcal{A}} = \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}}$*, it holds that* $\mathsf{Adv}^{\mathcal{R}^{\mathcal{A}}}_{\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}} \geq \epsilon_{\mathcal{R}}$ *and* $\mathcal{R}^{\mathcal{A}}$ *runs in time* $t_{\mathcal{R}}$*. Then there exists an algorithm* $\mathcal{M}$ *running in time* $t_{\mathcal{M}} \leq 2 \cdot t_{\mathcal{R}}$ *s.t.* $\mathsf{Adv}^{\mathcal{M}}_{\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}} \geq \epsilon_{\mathcal{R}} - \epsilon_{\mathcal{A}} \cdot \frac{2\exp(-1)}{q_{\mathsf{sk}}}$*.*

Theorem 5.4 implies that if there exists a reduction from $\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}$ to $\mathbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ for a signature scheme with rerandomizable keys RSig s.t. the reduction loses less than a factor proportional to q_{sk}, then there exists an efficient algorithm $\mathcal{M}$ that can break the $\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}$ security. We formulate and prove this result w.r.t. a reduction from the strongest possible security notion of signature schemes with rerandomizable keys (i.e., **uf-cma-rk**) to the restricted notion of one-per message wallet unforgeability (i.e., $\mathbf{wufcma1}_{\mathsf{HDWal}}$). Clearly, this implies that the result from Theorem 5.4 also holds for the weaker notion of **uf-cma-hrk1** for signature schemes with rerandomizable keys which we use in Theorem 5.3. We note that Theorem 5.4 can likewise be stated and proved with respect to the standard unforgeability notion of hierarchical deterministic wallet schemes, i.e., the notion that does not restrict the adversary to obtain at most one signature on a specific message.

6 Discussion

On Security Parameters We instantiate our generic hierarchical deterministic wallet construction $\mathsf{HDWal}[\mathsf{RSig}]$ with two schemes, namely $\mathsf{REC}[\mathsf{H}_1]$ (Figure 7) and $\mathsf{REC}'[\mathsf{H}_1]$ (Figure 6). Note that $\mathsf{HDWal}[\mathsf{REC}[\mathsf{H}_1]]$ corresponds to the BIP32 wallet, while $\mathsf{HDWal}[\mathsf{REC}'[\mathsf{H}_1]]$ is instantiated from the multiplicatively rerandomized construction $\mathsf{REC}'[\mathsf{H}_1]$ from [DFL19], we will refer to it as BIP32-m.

First, let us recall, how to compute the bit security level of a scheme. A hierarchical wallet scheme HDWal is said to have a bit security level of κ bits, if any algorithm $\mathcal{A}$ with running time t and advantage ϵ in $\mathbf{wufcma1}_{\mathsf{HDWal}}$ takes *expected* running time $\frac{t}{\epsilon} \geq 2^{\kappa}$ to break the scheme for the first time. (The security level for a conventional signature scheme is defined analogously). From our Theorems E.1, E.2, we compute the bit security level of our schemes, considering an algorithm $\mathcal{A}$ with parameters t', ϵ' (in game $\mathbf{wufcma1}_{\mathsf{HDWal}}$), where $t' \approx t$ and $\epsilon' = \epsilon \cdot Q$ for some $Q \geq 1$ and where t, ϵ denote the runtime and advantage of the related forger $\mathcal{C}$ in game $\mathbf{uf\text{-}cma1}_{\mathsf{EC}}$. By assumption, EC satisfies $\kappa = 128$ bits of security, hence $\frac{t}{\epsilon} \geq 2^{128}$. Thus, we obtain $\frac{t'}{\epsilon'} = \frac{t}{\epsilon \cdot Q} \geq \frac{2^{128}}{Q} = 2^{\kappa - \log Q}$. Our results are reported in Table 1, where we took an estimate of the practical parameters as follows: the total number of keys is $q = 2^{20}$, the number of q_{sk} of secret keys leaked is roughly 1% of the total number of keys q, i.e., $q_{\mathsf{sk}} \approx 2^{14}$.

Scheme	Theorem Ref.	Bit Security with $\kappa = 128$
BIP32	Thm E.1	$\log(Q) = \log(q \cdot 4e \cdot q_{\mathsf{sk}}) \approx 37, \kappa - \log(Q) = 91$
BIP32-m	Thm E.2	$\log(Q) = \log(4e \cdot q_{\mathsf{sk}}) \approx 17, \kappa - \log(Q) = 111$

Table 1: Bit Security Level of BIP32 and BIP32-m, relying on **uf-cma1** of $\mathsf{EC}[\mathsf{H}_0]$

On BIP32 Parameters. Our construction of $\mathsf{HDWal}[\mathsf{REC}[\mathsf{H}_1]]$ gives us the BIP32 construction as specified in [Wik18]. Here we list the exact parameters used in BIP32 and minor differences of BIP32 with our construction $\mathsf{HDWal}[\mathsf{REC}[\mathsf{H}_1]]$.

- Each node can derive at most 2^{32} children nodes.
- $\mathsf{e}_{\cdot}^{\cdot,\cdot}$ is chosen from $\{0,1\}^{32}$, which allows each non-hardened node to generate 2^{31} non-hardened and 2^{31} hardened child keys.
- A child key is derived as a hardened or a non-hardened node based on whether $\mathsf{e}_{\cdot}^{\cdot,\cdot} \geq 2^{31}$ or $\leq 2^{31}$ respectively. However, this is syntactical, and does not affect our security analysis.
- Although at each level, the total number of derived keys can be at most $(2^{32}) \cdot p$, where p is the number of parent nodes in the immediate upper level, we do not imagine that all of these keys are derived at every level. As can be seen, this would already exceed our parameter $q = 2^{20}$, as selected above.
- The chaincode $\mathsf{ch}_{\cdot,\cdot}$ is chosen from $\{0,1\}^{256}$.
- The input parameter $\mathbf{addr}_{\cdot,\cdot}$ to the key derivation algorithms is set to an empty string. We use this parameter to indicate the position in the tree, at which the child key is derived and to ensure that the actual BIP32 derivation algorithms are called on the proper inputs for this position.
- The input parameter $\mathbf{addr}_{\cdot,\cdot}$ to the key derivation algorithms is set to an empty string λ. Let us briefly explain this syntactical difference. In our Definition 4.2, $\mathbf{addr}_{\cdot,\cdot} \neq \lambda$ is provided as input. This makes the user aware of the position in the tree, at which the child key is derived and makes sure that the actual BIP32 derivation algorithms are called on the proper inputs for this position in the tree.

Open Questions Finally, let us mention some interesting open questions that can be answered in future works:

- Is it possible to remove the one per-message restriction and prove the security of the additively rerandomizable ECDSA scheme in the **uf-cma-hrk** notion? Additionally, is there a tight reduction to **uf-cma-hrk**?
- Can we improve the tightness of **uf-cma1** security [FKP17] of ECDSA from the semi-logarithm problem?

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable comments. This work is supported by the German Research Foundation (DFG) Emmy Noether Program FA 1320/1-1, by the German Research Foundation DFG - SFB 1119 - 236615297 (CROSSING Projects P1 and S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain Project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE.*

References

[ABFF09] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, and Keith B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12(3), January 2009. (Cited on page 5.)

[ADE$^+$20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20: 27th Conference on Computer and Communications Security*, pages 1017–1031, Virtual Event, USA, November 9–13, 2020. ACM Press. (Cited on page 5.)

[AGKK19] Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 426–445, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany. (Cited on page 5.)

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. (Cited on page 5.)

[BH19] Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 3–20, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany. (Cited on page 5.)

[Bit18] BitcoinExchangeGuide. CipherTrace Releases Report Exposing Close to $1 Billion Stolen in Crypto Hacks During 2018. https://coinexchangeguide.com/ciphertrace-releases-report-exposing-close-to-1-billion-stolen-in_-crypto-hacks-during-2018/, 2018. (Cited on page 1.)

[Blo18] Bloomberg. How to Steal $500 Million in Cryptocurrency. http://fortune.com/2018/01/31/coincheck-hack-how/, 2018. (Cited on page 1.)

[BR18] Michael Brengel and Christian Rossow. Identifying key leakage of bitcoin users. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 623–643, Cham, 2018. Springer International Publishing. (Cited on page 5.)

[But13] Vitalik Buterin. Deterministic Wallets, Their Advantages and their Understated Flaws. https://bitcoinmagazine.com/articles/deterministic-wallets-advantages-flaw-1385450276/, 2013. (Cited on page 2.)

[CEV14] Nicolas T. Courtois, Pinar Emirdag, and Filippo Valsorda. Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor RNG events. Cryptology ePrint Archive, Report 2014/848, 2014. https://eprint.iacr.org/2014/848. (Cited on page 5.)

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. (Cited on page 5, 18, 31, 32.)

[DFL19] Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 651–668. ACM Press, November 11–15, 2019. (Cited on page 2, 3, 4, 5, 6, 7, 8, 11, 18, 19, 33.)

[DKLs18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pages 980–997, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press. (Cited on page 5.)

[Ele13] Version bytes for BIP32 extended public and private keys. https://electrum.readthedocs.io/en/latest/xpub_version_bytes.html, 2013. (Cited on page 1.)

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 444–460, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. (Cited on page 5.)

[FKM$^+$16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 301–330, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany. (Cited on page 3, 5, 18, 30.)

[FKP16] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1651–1662, Vienna, Austria, October 24–28, 2016. ACM Press. (Cited on page 5.)

[FKP17] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the one-per-message unforgeability of (ec)dsa and its variants. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 519–534, Cham, 2017. Springer International Publishing. (Cited on page 4, 7, 19.)

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*, volume 9696 of *Lecture Notes in Computer Science*, pages 156–174, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany. (Cited on page 5.)

[GS15] Gus Gutoski and Douglas Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015: 19th International Conference on Financial Cryptography and Data Security*, volume 8975 of *Lecture Notes in Computer Science*, pages 497–504, San Juan, Puerto Rico, January 26–30, 2015. Springer, Heidelberg, Germany. (Cited on page 5.)

[KK18] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. *Journal of Cryptology*, 31(1):276–306, January 2018. (Cited on page 31.)

[KMOS19] Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh when you wake up: Proactive threshold wallets with offline devices. Cryptology ePrint Archive, Report 2019/1328, 2019. https://eprint.iacr.org/2019/1328. (Cited on page 5.)

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 33–61, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. (Cited on page 5.)

[Led14] Ledger Support,Ledger Nano OS. https://support.ledger.com/hc/en-us/articles/115005297709-Export-your-accounts, 2014. (Cited on page 1.)

[LFA20] Adriano Di Luzio, Danilo Francati, and Giuseppe Ateniese. Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20: 19th International Conference on Cryptology and Network Security*, volume 12579 of *Lecture Notes in Computer Science*, pages 323–343, Vienna, Austria, December 14–16, 2020. Springer, Heidelberg, Germany. (Cited on page 5.)

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1837–1854, Toronto, ON, Canada, October 15–19, 2018. ACM Press. (Cited on page 5.)

[MPs19] Antonio Marcedone, Rafael Pass, and abhi shelat. Minimizing trust in hardware wallets with two factor signatures. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 407–425, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany. (Cited on page 5.)

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. (Cited on page 5.)

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://ia.cr/2004/332. (Cited on page 6.)

[Ske18] Rhys Skellern. Cryptocurrency Hacks: More Than $2b USD lost between 2011-2018. https://medium.com/ecomi/cryptocurrency-hacks-more-than-2b-usd-lost-between-2011-2018_-67054b342219, 2018. (Cited on page 1.)

[Tre14] Trezor Wiki,Cryptocurrency standards,Hierachical deterministic wallets. https://wiki.trezor.io/Cryptocurrency_standards, 2014. (Cited on page 1.)

[TVR16] Mathieu Turuani, Thomas Voegtlin, and Michaël Rusinowitch. Automated verification of electrum wallet. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 27–42, Christ Church, Barbados, February 26, 2016. Springer, Heidelberg, Germany. (Cited on page 5.)

[Wik18] Bitcoin Wiki. BIP32 proposal. https://en.bitcoin.it/wiki/BIP_0032, 2018. (Cited on page 1, 10, 19.)

[ZCC$^+$15] Zongyang Zhang, Yu Chen, Sherman S. M. Chow, Goichiro Hanaoka, Zhenfu Cao, and Yunlei Zhao. Black-box separations of hash-and-sign signatures in the non-programmable random oracle model. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages

435–454, Kanazawa, Japan, November 24–26, 2015. Springer, Heidelberg, Germany. (Cited on page 5.)

A Proof of Theorem 3.3

Proof. For this proof, we consider an adversary $\mathcal{A}$ playing in the $\textbf{uf-cma-hrk1}_{\mathsf{REC}[\mathsf{H}_1]}$ game relative to a random oracle H_1. Below, we present a series of games $\boldsymbol{G}_0$ to $\boldsymbol{G}_6$ where the following holds.

$$\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk1}_{\mathsf{REC}[\mathsf{H}_1]}} = \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G_6}^{\mathcal{A}} = 1] + \frac{{q_{\mathsf{H}_1}}^2}{p}$$

Game $\boldsymbol{G_0}$: This game is equivalent to the original game, namely $\textbf{uf-cma-hrk1}^{\mathcal{A}}_{\mathsf{REC}[\mathsf{H}_1]}$. In particular, a key pair $(\mathsf{sk}, \mathsf{pk})$ is sampled as $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{REC}[\mathsf{H}_1].\mathsf{Gen}(\mathsf{par})$. The adversary $\mathcal{A}$ is given pk as the challenge public key and oracle access to `Rand`, `RSign` and random oracle H_1. $\mathcal{A}$ can query `Rand` to receive a randomness ρ and make a follow-up query to `RSign` to receive a signature on message m with respect to the rerandomized key $\mathsf{pk}' \leftarrow \mathsf{pk} + \rho \cdot G$. In particular, $\mathcal{A}$ is allowed to query `RSign` on every input pair (m, ρ) at most once. Additionally, $\mathcal{A}$ can make direct queries to the random oracle H_1. Eventually, in order to win the game, $\mathcal{A}$ has to come up with a valid forgery σ^* on a new message m^* with respect to a randomness ρ^*. Since $\boldsymbol{G_0}$ proceeds as **uf-cma-hrk1** we have that $\Pr[\boldsymbol{G_0}^{\mathcal{A}} = 1] = \Pr[\textbf{uf-cma-hrk1}^{\mathcal{A}}_{\mathsf{REC}[\mathsf{H}_1]} = 1] = \mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk1}_{\mathsf{REC}[\mathsf{H}_1]}}$.

Game $\boldsymbol{G_1}$: This game is similar to game $\boldsymbol{G_0}$ with the following modification. $\mathcal{A}$ is now given a public key $\widetilde{\mathsf{pk}}$ instead of pk (which served as the challenge public key in $\boldsymbol{G}_0$) as the challenge public key. $\widetilde{\mathsf{pk}}$ is derived as $\widetilde{\mathsf{pk}} \leftarrow \mathsf{pk} - \tilde{\rho} \cdot G$ with a freshly sampled randomness $\tilde{\rho} \xleftarrow{\$} \mathcal{R}$. The corresponding secret key is obtained as $\widetilde{\mathsf{sk}} = \mathsf{sk} - \tilde{\rho}$.

Due to the perfect rerandomizablity of keys of the rerandomizable signature scheme REC, pk is indistinguishable from $\widetilde{\mathsf{pk}}$. Hence, we have $\Pr[\boldsymbol{G_0}^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G_1}^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G_2}$: This game is similar to game $\boldsymbol{G_1}$ with the following modification in the `Rand` oracle. An index j is sampled uniformly at random from the set $\{1, \ldots, q\}$, where q is an upper bound on the number of queries to the oracle `Rand`. The game returns $\tilde{\rho}$ at the j^{th} `Rand` query. For all other queries, ρ is sampled randomly as $\rho \xleftarrow{\$} \mathcal{R}$.

Since both $\tilde{\rho}$ and ρ are sampled randomly from $\mathcal{R}$, the output distribution of the `Rand` oracle is the same in games $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$. Hence, we have $\Pr[\boldsymbol{G_2}^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G_1}^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G_3}$: This game behaves exactly like the game $\boldsymbol{G_2}$ with the following modifications: First, the game internally maintains a random oracle H_0 (in addition to H_1) in a straightforward manner, by storing a list H_0 of query/response pairs. Second, the game programs the oracle H_1 by maintaining three lists H_1, H_1' and Γ, where the first two will be used as possible replies to queries to H_1, and Γ stores pre-computed signatures. In the beginning of the game, H_1, H_1' and Γ are initially set to $\perp$ in each entry. Whenever $\mathcal{A}$ queries a message m to H_1, the values $H_1[m]$, $H_1'[m]$ and $\Gamma[m]$ are set in one of two ways depending on whether m is prefixed with a public key pk' or not. Here, pk' is a rerandomized form of the public key $\widetilde{\mathsf{pk}}$ (i.e., $\mathsf{pk}' \leftarrow \widetilde{\mathsf{pk}} + \rho \cdot G$ where $\rho \leftarrow$ `Rand` is a previous answer to any `Rand` oracle query), where $\widetilde{\mathsf{pk}} = \mathsf{pk} - \tilde{\rho} \cdot G$ (see Game $\boldsymbol{G}_1$). Concretely, on query m to H_1, the lists H_1, H_1' and Γ are maintained in the following way:

- If H_1 has already been programmed in a previous query, i.e., $H_1[m] \neq \perp$, return $H_1[m]$.
- Else $H_1[m] = \perp$, then sample uniformly at random $h \xleftarrow{\$} \mathbb{Z}_p$, set $H_1[m] = h$, and proceed as follows:
  - Case 1: m is of the form (pk', m'), where $\mathsf{pk}' = \widetilde{\mathsf{pk}} + \rho \cdot G = \mathsf{pk} + (\rho - \tilde{\rho}) \cdot G$, for $\rho \in \mathsf{RList}$. Derive a signature σ as $\sigma \leftarrow \mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}', m')$ for $\mathsf{sk}' = \widetilde{\mathsf{sk}} + \rho = \mathsf{sk} + (\rho - \tilde{\rho}) \pmod p$ and parse $\sigma := (r, s)$. Then set $H_1'[m] = H_0[m] - r \cdot (\rho - \tilde{\rho}) \pmod p$ and $\Gamma[m] = \sigma$. Finally return $H_1[m]$.
  - Case 2: m is not of the form (pk', m'). Set $\Gamma[m] = \epsilon$ and return $H_1[m]$.

In both the cases, the output of H_1 is uniformly distributed from $\mathcal{A}$'s point of view. It follows that $\Pr[\boldsymbol{G_2}^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G_3}^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G_4}$: This game proceeds as the previous game with a modification in the Rand oracle. Upon $\mathcal{A}$ querying the Rand oracle, sample ρ as before, then compute the rerandomized public key $\mathsf{pk}' \leftarrow \widetilde{\mathsf{pk}} + \rho \cdot G$ and check if there exists a message m with prefix pk' such that $\Gamma[m] = \epsilon$. In that case, the game aborts.

Claim A.1 Let E_1 be the event that the game $\boldsymbol{G_4}$ aborts during a Rand query. Then, we have that $\Pr[\mathsf{E}_1] \leq \frac{{q_{\mathsf{H}_1}}^2}{p}$.

Proof. Event E_1 can only occur if $\mathcal{A}$ has queried H_1 on input m with prefix $\mathsf{pk}' \leftarrow \widetilde{\mathsf{pk}} + \rho \cdot G$ prior to making a query to Rand that returns ρ. Since $\mathcal{A}$ makes at most q_{H_1} queries to H_1, for each query to Rand that the adversary $\mathcal{A}$ makes, we have that with probability $\frac{q_{\mathsf{H}_1}}{p}$ we receive a value ρ such that $\mathsf{pk}' \leftarrow \widetilde{\mathsf{pk}} + \rho \cdot G$ is a prefix of input m that was earlier made to H_1. Since there are at most q such queries to Rand by taking the union bound over q_{H_1} we obtain $\Pr[\mathsf{E}_1] = \sum_{i=1}^{q_{\mathsf{H}_1}} \frac{q_{\mathsf{H}_1}}{p} = \frac{{q_{\mathsf{H}_1}}^2}{p}$. ■

From the above, we have that $\Pr[\boldsymbol{G_3}^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G_4}^{\mathcal{A}} = 1] + \frac{{q_{\mathsf{H}_1}}^2}{p}$.

Game $\boldsymbol{G_5}$: This game is similar to the game $\boldsymbol{G_4}$ except for a modification in the RSign oracle. Upon $\mathcal{A}'s$ query on input (m, ρ), the game simulates the RSign oracle in the following manner. It computes the rerandomized public key $\mathsf{pk}' \leftarrow \widetilde{\mathsf{pk}} + \rho \cdot G$ and creates the public key prefixed message $\mathsf{pm} \leftarrow (\mathsf{pk}', m)$. The signature is implicitly derived via querying the simulated random oracle H_1 (see Game $\boldsymbol{G_3}$ above) on input the prefixed message pm. This results into $\Gamma[\mathsf{pm}] = \sigma = \mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}', m)$, which is returned as the response to the signature query.

Observe that all queries to RSign on input the tuple (m, ρ) output the same signature. However, since ECDSA signatures are randomized, the output of RSign should be different with overwhelming probability for each query on the same input tuples. Here, we exploit that $\mathcal{A}$ is allowed to query RSign at most once for the same input pair (m, ρ). Hence, the output distribution of RSign is identical to the distribution of the RSign oracle in the previous game and it holds that $\Pr[\boldsymbol{G_4}^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G_5}^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G_6}$: This game is similar to game $\boldsymbol{G_5}$ except for the following changes: In the oracles RSign and H_1 the game uses $\mathsf{EC}[H_0].\mathsf{Sign}$ instead of $\mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}$ to compute the signatures stored in Γ (and in case of RSign this implicitly happens via H_1). More precisely, when H_1 is queried on $\mathsf{pm} = (\mathsf{pk}', m')$, where $\mathsf{pk}' = \widetilde{\mathsf{pk}} + \rho \cdot G = \mathsf{pk} + (\rho - \tilde{\rho}) \cdot G$ for $\rho \in \mathsf{RList}$, we derive $\sigma \leftarrow \mathsf{EC}[H_0].\mathsf{Sign}(\mathsf{sk}, \mathsf{pm})$, for $\mathsf{sk}' = \mathsf{sk} + (\rho - \tilde{\rho}) \pmod p$. Furthermore, upon H_1 being queried on m, H_1 returns $H_1'[m]$ instead of $H_1[m]$ whenever $\Gamma[m] \neq \perp$ and $\Gamma[m] \neq \epsilon$.

Claim A.2 It holds that $\Pr[\boldsymbol{G_5}^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G_6}^{\mathcal{A}} = 1]$.

Proof. First, note that in this game, H_1 returns $H_0[m] - r \cdot (\rho - \tilde{\rho})$ on a message m for which a signature is stored in Γ. We have to show now that when H_1 is queried on $\mathsf{pm} = (\mathsf{pk}', m')$, where $\mathsf{pk}' = \mathsf{pk} + (\rho - \tilde{\rho}) \cdot G$ and $\mathsf{sk}' = \mathsf{sk} + (\rho - \tilde{\rho}) \pmod p$ for $\rho \in \mathsf{RList}$, we derive $\sigma \leftarrow \mathsf{EC}[H_0].\mathsf{Sign}(\mathsf{sk}, \mathsf{pm})$ (Game $\boldsymbol{G_6}$) instead of computing $\sigma \leftarrow \mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}', m')$ (Game $\boldsymbol{G_5}$).

To this end, we recall Lemma 3.2, which states that if $\sigma = (r, s)$ is a valid signature for $\mathsf{pm} \leftarrow (\mathsf{pk}', m')$ under pk w.r.t. $\mathsf{EC}[H_0]$, it is also a valid signature for m' under $\mathsf{pk}' \leftarrow \mathsf{pk} + (\rho - \tilde{\rho}) \cdot G$ w.r.t. $\mathsf{REC}[\mathsf{H}_1]$, if it holds that $\mathsf{H}_1(\mathsf{pm}) = \mathsf{H}_0(\mathsf{pm}) - r \cdot (\rho - \tilde{\rho}) \pmod p$. Note that we replaced the $\mathsf{REC}[\mathsf{H}_1].\mathsf{Sign}$ procedure call on a message m' in $\boldsymbol{G_5}$ by a $\mathsf{EC}[\mathsf{H}_0].\mathsf{Sign}$ procedure call on a prefixed message $\mathsf{pm} \leftarrow (\mathsf{pk}', m')$, where $\mathsf{pk}' = \mathsf{pk} + (\rho - \tilde{\rho}) \cdot G$. It remains to show that the condition $\mathsf{H}_1(\mathsf{pm}) = \mathsf{H}_0(\mathsf{pm}) - r \cdot (\rho - \tilde{\rho}) \pmod p$ holds. But since $H_1'[\mathsf{pm}] = H_0[\mathsf{pm}] - r \cdot (\rho - \tilde{\rho}) \pmod p$ is programmed accordingly (latest when RSign is queried), this follows directly. ■

Combining results from $\boldsymbol{G}_0$ to $\boldsymbol{G}_6$, we have that

$$\Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] + \frac{{q_{\mathsf{H}_1}}^2}{p}. \tag{1}$$

Reduction to uf-cma1 security. Having shown that the original $\textbf{uf-cma-hrk1}^{\mathcal{A}}_{\mathsf{REC}[\mathsf{H}_1]}$ game is indistinguishable from game $\boldsymbol{G_6}$, it remains to show that an adversary $\mathcal{A}$ winning in game $\boldsymbol{G}_6$ can be turned into an adversary $\mathcal{C}$ that wins $\textbf{uf-cma1}^{\mathcal{C}}_{\mathsf{EC}[\mathsf{H}_0]}$ game with related success probability. To this end, we construct $\mathcal{C}$ that runs in the game $\textbf{uf-cma1}^{\mathcal{C}}_{\mathsf{EC}[\mathsf{H}_0]}$ and simulates to $\mathcal{A}$ game $\boldsymbol{G}_6$. Thus, $\mathcal{C}$ proceeds as game $\boldsymbol{G}_6$ and leverages oracle access to its own signing oracle (with respect to its challenge public key) in the following way:

1. On input the challenge public key pk_C from $\mathbf{uf\text{-}cma1}_{\mathsf{EC}[\mathsf{H}_0]}$, the adversary $\mathcal{C}$ sets pk to $\mathsf{pk}_{\mathcal{C}}$. Note that this implicitly sets the challenge public key in $\mathcal{C}$'s simulation of $\boldsymbol{G}_6$ to $\widetilde{\mathsf{pk}} = \mathsf{pk}_{\mathcal{C}} - \tilde{\rho} \cdot G$. Hence, $\mathcal{C}$ runs $\mathcal{A}$ on input $\widetilde{\mathsf{pk}}$.

2. In case $\mathcal{A}$ returns a forgery (m^*, σ^*, ρ^*) with $\rho^* \neq \tilde{\rho}$, $\mathcal{C}$ aborts.

$\mathcal{C}$ perfectly simulates $\boldsymbol{G}_6$ for $\mathcal{A}$ except in case where it aborts. Moreover, note that in case there is no abort, we have that

$$\mathsf{pk}^* = \widetilde{\mathsf{pk}} + \rho^* \cdot G = \mathsf{pk}_{\mathcal{C}} - \tilde{\rho} \cdot G + \tilde{\rho} \cdot G = \mathsf{pk}_{\mathcal{C}}.$$

From the above programming strategy, we conclude that for $\mathcal{A}$'s queries to H_1 that are prefixed with pk^*, the oracles H_0 and H_1 are identical. It remains to calculate the success probability of $\mathcal{C}$ in winnning the $\mathbf{uf\text{-}cma1}_{\mathsf{EC}[\mathsf{H}_0]}$ game in case $\mathcal{A}$ returns a valid forgery.

Claim A.3 Let E_2 be the event that $\mathcal{A}$ outputs (m^*, σ^*, ρ^*) s.t. $(\mathsf{pm}^*, \sigma^*)$ constitutes a valid forgery in game $\mathbf{uf\text{-}cma1}^{\mathcal{C}}_{\mathsf{EC}[\mathsf{H}_0]}$. Then, we have that $\Pr[\mathsf{E}_2 | \boldsymbol{G_6}^{\mathcal{A}} = 1] \geq \frac{1}{q}$, where q is the number of queries to the `Rand` oracle.

Proof. In order to prove this claim, we need to show that with probability $\frac{1}{q}$ it must hold that (1) $(\mathsf{pm}^*, \sigma^*)$ is a valid forgery in game $\mathbf{uf\text{-}cma1}^{\mathcal{C}}_{\mathsf{EC}[\mathsf{H}_0]}$ under public key $\mathsf{pk}_{\mathcal{C}}$ and (2) the `Sign` oracle of the $\mathbf{uf\text{-}cma1}^{\mathcal{C}}_{\mathsf{EC}[\mathsf{H}_0]}$ game has not been queried on input pm^*.
First, note that if σ^* is a valid signature for message (pk^*, m^*) under the public key pk^* relative to $\mathsf{REC}[\mathsf{H}_1]$, then σ^* is also a valid signature on pm^* under public key $\mathsf{pk}_{\mathcal{C}} = \mathsf{pk}^*$ relative to $\mathsf{EC}[\mathsf{H}_0]$, as H_0 and H_1 are identical for messages prefixed with pk^*. Since there are at most q possible values of ρ^* and $\mathcal{C}$ chooses one of them uniformly at random, the probability that $\mathcal{C}$'s guess is correct is at least $\frac{1}{q}$. Note that from the adversary's perspective, the public key generated at index j is no different than other public keys.
Second, since (m^*, σ^*, ρ^*) is a valid forgery in $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{A}}_{\mathsf{REC}[\mathsf{H}_1]}$, $\mathcal{A}$ has not previously queried the `RSign` oracle on input (m^*, ρ^*). Correspondingly, the `Sign` oracle of the $\mathbf{uf\text{-}cma1}_{\mathsf{EC}[\mathsf{H}_0]}$ game has also not been queried on message pm^* and hence, $(\mathsf{pm}^*, \sigma^*)$ is a valid forgery in $\mathbf{uf\text{-}cma1}_{\mathsf{EC}[\mathsf{H}_0]}$.

■

From Eq. 1 we get the following.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{REC}[\mathsf{H}_1]}} = \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] + \frac{q^2_{\mathsf{H}_1}}{p}$$
$$\text{or, } \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] \geq \mathsf{Adv}^{\mathcal{A}}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{REC}[\mathsf{H}_1]}} - \frac{q^2_{\mathsf{H}_1}}{p}$$

Since $\mathcal{C}$ can use a valid forgery by $\mathcal{A}$ in its own game whenever E_2 occurs,

$$\mathsf{Adv}^{\mathcal{C}}_{\mathbf{uf\text{-}cma}_{\mathsf{EC}}[\mathsf{H}_0]} \geq \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] \cdot \Pr[\mathsf{E}_2 \mid \boldsymbol{G}_6^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] \cdot \frac{1}{q}$$
$$\geq \left(\mathsf{Adv}^{\mathcal{A}}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{REC}[\mathsf{H}_1]}} - \frac{q^2_{\mathsf{H}_1}}{p} \right) \cdot \frac{1}{q}$$

■

B Unlinkability Proof of Generic Construction

Theorem B.1 *Let* $\mathsf{HDWal}[\mathsf{RSig}]$ *be the construction defined in Figure 5. Then for any adversary* $\mathcal{A}$ *playing in game* $\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal}[\mathsf{RSig}]}$ *there exists an adversary* $\mathcal{A}_1$ *that plays in the game* $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathbf{unl}_{\mathsf{HDWal}[\mathsf{RSig}]}} \leq \frac{q_{\mathsf{H}}(q_C + 1)}{2^{\kappa}} + \mathsf{Adv}^{\mathcal{A}_1}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}}$$

where q_{H} *and* q_C *are the number of random oracle and child creation queries from* $\mathcal{A}$*, respectively.*

Proof. Consider the $\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ game for an adversary $\mathcal{A}$. In the beginning, the challenger generates a fresh master key pair and chaincode

$$(\mathsf{msk}_{\mathsf{nh},0,0}, \mathsf{mpk}_{\mathsf{nh},0,0}, \mathsf{ch}_{0,0}) \leftarrow \mathsf{HDWal[RSig].Setup(par)}$$

and runs $\mathcal{A}$ on inputs the security parameter and the master public key $\mathsf{mpk}_{\mathsf{nh},0,0}$. During the output phase of the $\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ game, $\mathcal{A}$ outputs a tuple $(\mathbf{addr}_{i,s}, \mathsf{e}^{s,t}_{i+1}, c)$, where $\mathsf{e}^{s,t}_{i+1}$ is the edge from the node with address $\mathbf{addr}_{i,s}$ to the challenge node with address $\mathbf{addr}_{i+1,t}$ and c indicates if $\mathbf{addr}_{i+1,t}$ is a hardened or non-hardened node. In case $\mathbf{addr}_{i,s}$ is a hardened node, the game aborts and hence we have that the adversary's advantage is 0, i.e., $\mathsf{Adv}^{\mathcal{A}}_{\mathbf{unl}_{\mathsf{HDWal[RSig]}}} = 0$. Likewise, if $\mathcal{A}$ has previously queried the `CHLeakO` oracle on address $\mathbf{addr}_{i,s}$ or any of its prefix addresses, i.e., $\mathsf{CH}[\mathbf{addr}^{j}_{i,s}] = 1$ for any $j < i$, and if the challenge node is non-hardened (i.e., $c = \mathsf{nh}$) then the game aborts and we have that $\mathsf{Adv}^{\mathcal{A}}_{\mathbf{unl}_{\mathsf{HDWal[RSig]}}} = 0$.

Let $\mathsf{pk}_{\mathsf{nh},i+1,t}$ and $\mathsf{pk}_{\mathsf{nh},1,t}$ denote the challenge public keys in case $\mathcal{A}$ challenges a non-hardened node (i.e., $c = \mathsf{nh}$) with address $\mathbf{addr}_{i+1,t}$. Further, let $(\mathsf{pk}_{j,\cdot}, \mathsf{ch}_{j,\cdot})$ for $1 \leq j \leq i$ denote the public key and chaincode pair of all nodes in the prefix address of $\mathbf{addr}_{i+1,t}$. Recall that the non-hardened public keys are computed as follows:

$$\begin{aligned}(\omega, \mathsf{ch}_{j+1,t}) \leftarrow & \mathsf{H}(\mathsf{pk}_{j,s}, \mathsf{ch}_{j,s}, \mathsf{e}^{s,t}_{j+1}),\\ \mathsf{pk}_{j+1,t} \leftarrow & \mathsf{RSig.RandPK}(\mathsf{pk}_{j,s}; \omega)\end{aligned}$$

According to the (perfect) rerandomizability of keys property (cf. Def 2.2) the public keys derived via the $\mathsf{RSig.RandPK}$ algorithm are identically distributed to freshly generated keys from $\mathcal{A}$'s view as long as ω is uniformly random. Therefore, the challenge public keys $\mathsf{pk}_{\mathsf{nh},i+1,t}$ and $\mathsf{pk}_{\mathsf{nh},1,t}$ are identically distributed from $\mathcal{A}$'s point of view as long as $\mathcal{A}$ has not previously queried the random oracle H on input $(\mathsf{pk}_{j,\cdot}, \mathsf{ch}_{j,\cdot}, \mathsf{e}^{\cdot,\cdot}_{j+1})$. If $\mathcal{A}$ makes one of the aforementioned queries, it can recursively compute the public key of the challenge node, thereby trivially winning the $\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ game. By assumption, $\mathcal{A}$ makes at most q_C queries to the child creation oracles. Therefore, there are at most $q_C + 1$ potential chaincodes that $\mathcal{A}$ can guess correctly and query the random oracle on. For each of these, the probability of correctly guessing it is $\frac{1}{2^\kappa}$ and thereby the probability of correctly guessing any of the chaincodes is at most $\frac{q_C+1}{2^\kappa}$ during any particular random oracle query. Since $\mathcal{A}$ makes at most q_{H} calls to H, the overall probability of querying the random oracle on an input as above is $\frac{q_{\mathsf{H}}(q_C+1)}{2^\kappa}$.

It remains to show $\mathcal{A}$'s probability of winning the $\mathbf{unl}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ game in case the adversary challenges a hardened node with address $\mathbf{addr}_{i+1,t}$. In this case, let $\mathsf{pk}_{\mathsf{h},i+1,t}$ and $\mathsf{pk}_{\mathsf{h},1,t}$ denote the challenge public keys and let $(\mathsf{pk}_{j,\cdot}, \mathsf{ch}_{j,\cdot})$ for $1 \leq j \leq i$ denote the public key and chaincode pair of all nodes in the prefix address of $\mathbf{addr}_{i+1,t}$.

$\mathcal{A}$ is allowed to query the `CHLeakO` oracle for parent nodes, thereby eliminating the need to correctly guess a relevant chaincode. Recall that hardened public keys are derived as follows:

$$\begin{aligned}(\omega, \mathsf{ch}_{j+1,t}) \leftarrow & \mathsf{H}(\mathsf{sk}_{j,s}, \mathsf{ch}_{j,s}, \mathsf{e}^{s,t}_{j+1})\\ \mathsf{pk}_{j+1,t} \leftarrow & \mathsf{RSig.RandPK}(\mathsf{pk}_{j,s}; \omega)\end{aligned}$$

Hence, having access to the `CHLeakO` oracle does not reveal all required inputs to the random oracle, i.e., the secret key of the parent node is still unknown to the adversary. As such, according to the (perfect) rerandomizability of keys property (cf. Def 2.2), $\mathcal{A}$ can distinguish $\mathsf{pk}_{\mathsf{h},i+1,t}$ from $\mathsf{pk}_{\mathsf{h},1,t}$ only if it is able to compute the secret key of one of the challenge nodes' parents. Let E be the event that $\mathcal{A}$ can compute a secret key $\mathsf{sk}_{j,\cdot}$ that corresponds to any of the public keys $\mathsf{pk}_{j,\cdot}$ and calls the random oracle on input $(\mathsf{sk}_{j,\cdot}, \cdot, \cdot)$. Then we can upper bound the probability that event E occurs as follows:

Claim B.2 There exists an algorithm $\mathcal{A}_1$ such that

$$\mathsf{Adv}^{\mathcal{A}_1}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} \geq \Pr[\mathsf{E}].$$

Proof. The proof of this claim corresponds to the proof of claim C.1 in Appendix C. ■

Therefore, the adversary's advantage in case of a hardened challenge node can be upper bounded by $\mathsf{Adv}^{\mathcal{A}_1}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}}$ and $\mathcal{A}$'s overall advantage in game $\textbf{unl}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ can be upper bounded by $\mathsf{Adv}^{\mathcal{A}}_{\textbf{unl}_{\mathsf{HDWal[RSig]}}} \leq \frac{q_{\mathsf{H}}(q_C+1)}{2^\kappa} + \mathsf{Adv}^{\mathcal{A}_1}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}}$.

■

Indistinguishability of Hardened nodes Recall that in our construction $\mathsf{HDWal[RSig]}$, a hardened key pair $(\mathsf{sk}_{\mathsf{h},(i+1),t}, \mathsf{pk}_{\mathsf{h},(i+1),t})$ is derived via $\mathsf{SKDer}_\mathsf{H}$ and $\mathsf{PKDer}_\mathsf{H}$ as follows:

$$\begin{aligned}(\omega, \mathsf{ch}_{(i+1),t}) &\leftarrow \mathsf{H}(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{ch}_{i,s}, \mathsf{e}^{s,t}_{i+1})\\ \mathsf{sk}_{\mathsf{h},(i+1),t} &\leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}_{\mathsf{nh},i,s}; \omega)\\ \mathsf{pk}_{\mathsf{h},(i+1),t} &\leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}_{\mathsf{nh},i,s}; \omega)\end{aligned}$$

Due to the key rerandomizability property of the underlying signature scheme RSig, $\mathcal{A}$ can only distinguish $(\mathsf{sk}_{\mathsf{h},(i+1),t}, \mathsf{pk}_{\mathsf{h},(i+1),t})$ from a fresh key pair if it can distinguish ω from random. Since we model H as a random oracle, this happens only if $\mathcal{A}$ has previously queried H on the same input, i.e., $(\mathsf{sk}_{\mathsf{nh},i,s}, \mathsf{St}_{i,s}, \mathsf{e}_{i+1,t})$. Since our model excludes secret key leakage of non-hardened nodes, the adversary cannot distinguish the output of H from a random value except if it correctly guesses $\mathsf{sk}_{\mathsf{nh},i,s}$ or any parent secret key that $\mathsf{sk}_{\mathsf{nh},i,s}$ has been (directly or indirectly) derived from.

C Proof of Theorem 5.3

Proof. Before we give the full formal proof of Theorem 5.3, we first provide a high level overview of the proof. We show that the generic construction $\mathsf{HDWal[RSig]}$ is one-per message unforgeable w.r.t. game $\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}$, if the signature scheme with rerandomizable keys RSig is one-per message unforgeable w.r.t. the game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$. The main idea behind the proof is as follows. First, the adversary $\mathcal{C}$ in game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$ receives a public key $\mathsf{pk}_{\mathcal{C}}$. It chooses a random chaincode and uses it to derive a key mpk, which it embeds mpk as the master public key for $\mathcal{A}$ in game $\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}$. Note that these changes cannot be detected by $\mathcal{A}$ due to the rerandomizability of keys and the transitivity property of the RSig scheme (see Definitions 2.2 and 5.1). Second, $\mathcal{C}$ attempts to predict, for each hardened node in the tree, with a certain probability if this node will get corrupted by $\mathcal{A}$ throughout the game. For these nodes, $\mathcal{C}$ generates a fresh key pair independently of $\mathsf{pk}_{\mathcal{C}}$. For the other hardened nodes, $\mathcal{C}$ derives non-hardened public keys (instead of hardened public keys) from $\mathsf{pk}_{\mathcal{C}}$. It is crucial here that the non-hardened public keys are derived from $\mathsf{pk}_{\mathcal{C}}$ instead of from the parent public key, since $\mathsf{pk}_{\mathcal{C}}$ is not known to the adversary and therefore $\mathcal{A}$ is not able to distinguish the non-hardened public key from a hardened public key. We show that this guessing introduces a polynomial loss in the number of corrupted hardened nodes for $\mathcal{C}$'s advantage to win game $\textbf{uf-cma-hrk}_{\mathsf{RSig}}$ game but that $\mathcal{C}$ still wins the game with non-negligible probability.

We now provide the formal proof via a series of games $\boldsymbol{G}_0$ to $\boldsymbol{G}_6$.

Game $\boldsymbol{G}_0$: This is the regular $\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}$ game at the beginning of which a key pair $(\mathsf{pk}, \mathsf{sk})$ is generated and the adversary $\mathcal{A}$ is given as input pk and oracle access to the following oracles: `HChildO`, `NHChildO`, `HDSignO`, `CHLeakO` and `SKLeakO` oracles and a random oracle H. The random oracle H is internally programmed in a straight forward manner, by maintaining a list H. In particular, on input s, if $H[s] \neq \perp$, then return $H[s]$. Otherwise, sample a fresh randomness $\rho \xleftarrow{\$} \mathcal{R}$ and a fresh value as $\psi \xleftarrow{\$} \{0,1\}^\kappa$ and set $(\rho, \psi) =: H[s]$ and return $H[s]$. In addition, the game keeps a list R in which it stores the randomness used to derive the keys at position s and level i at entry $R[i,s]$. We have that $\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}} = \Pr[\textbf{wufcma1}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}} = 1] = \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_1$: Upon generating the key pair $(\mathsf{pk}, \mathsf{sk})$, the game chooses a fresh chaincode $\mathsf{ch}_{0,0} \xleftarrow{\$} \{0,1\}^\kappa$ and fresh randomness $\rho \xleftarrow{\$} \mathcal{R}$. Then it derives the root public key for the $\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}$ game as $\mathsf{mpk}_{0,0} \xleftarrow{\$} \mathsf{RSig.RandPK}(\mathsf{pk}; \rho)$, stores ρ in a list as $R[0,0] = \rho$. The game sends $\mathsf{ch}_{0,0}$ and $\mathsf{mpk}_{0,0}$ to $\mathcal{A}$.

Since the randomness ρ is chosen uniformly at random from $\mathcal{R}$, the rerandomizability of keys property of the signature scheme RSig holds. This implies that the distributions of $(\cdot, \mathsf{mpk}_{0,0})$ and

$(\cdot, \mathsf{mpk}'_{0,0}) \xleftarrow{\$} \mathsf{RSig.Gen}(\mathsf{par})$ are identical. Therefore, it holds that $\Pr[\boldsymbol{G}_1^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_2$: This game behaves like $\boldsymbol{G}_1$ with a modification in the `NHChildO` oracle. Upon an oracle query on input $(\mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$ the `NHChildO` oracle executes $\mathsf{PKDer}_{\mathsf{NH}}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$ and creates the public key $\mathsf{pk}_{\mathsf{nh},i+1,t}$ at level $i+1$ and position t as $\mathsf{pk}_{\mathsf{nh},i+1,t} \leftarrow \mathsf{RandPK}(\mathsf{pk}; \omega \odot R[i,s])$, i.e., the public key $\mathsf{pk}_{\mathsf{nh},i+1,t}$ is derived directly from pk with randomness $\omega \odot R[i,s]$, where $(\omega, \cdot) \leftarrow \mathsf{H}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathsf{e}_{i+1}^{s,t})$. The game then sets the list $R[i+1,t] = \omega \odot R[i,s]$. If any of the values $(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, R[i,s])$ is not defined yet, the game recursively derives the path from the root node up to $(\mathsf{pk}_{i,s}, \mathbf{addr}_{i,s})$ and updates the list up to $R[i,s]$.

Note that $\mathsf{RandPK}(\mathsf{pk}; \omega \odot R[i,s])$ and $\mathsf{RandPK}(\mathsf{pk}_{\mathsf{nh},i,s}; \omega)$ derive the same key $\mathsf{pk}_{\mathsf{nh},i+1,t}$, due to the transitive property of rerandomizable keys. Since ω and $R[i,s]$ are uniformly at random from $\mathcal{R}$, we have that $\Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_1^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_3$: This game proceeds similarly to the previous game with a modification in the random oracle. The game aborts upon the adversary querying the random oracle on input $(\mathsf{sk}_{\mathsf{nh},i,s}, \cdot, \cdot)$ where $\mathsf{sk}_{\mathsf{nh},i,s}$ is either a non-hardened secret key that corresponds to a public key $\mathsf{pk}_{\mathsf{nh},i,s}$ previously output by the `NHChildO` oracle or $\mathsf{sk}_{\mathsf{nh},i,s}$ is the master secret key $\mathsf{msk}_{0,0}$ corresponding to $\mathsf{mpk}_{0,0}$.

Claim C.1 Let ϵ be the probability that game $\boldsymbol{G}_3$ aborts during a random oracle query. Then there exists an algorithm $\mathcal{C}_1$ playing in game $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ such that $\mathsf{Adv}^{\mathcal{C}_1}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} \geq \epsilon$.

Proof. We prove this claim by providing a reduction to the **uf-cma-hrk1** security of RSig. More concretely, we show that there exists an algorithm $\mathcal{C}_1$ with $\mathsf{Adv}^{\mathcal{C}_1}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} \geq \epsilon$ assuming $\mathcal{C}_1$ has access to an adversary $\mathcal{A}$ that causes $\boldsymbol{G}_3$ to abort with probability ϵ. Initially, $\mathcal{C}_1$ receives as input a public key pk from the $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ game and chooses at random a chaincode $\mathsf{ch} \xleftarrow{\$} \{0,1\}^{\kappa}$. From pk and ch, $\mathcal{C}_1$ can honestly simulate the `NHChildO` and `CHLeakO` oracles to $\mathcal{A}$. The simulation of the random oracle H works as described in $\boldsymbol{G}_3$ with the exception that instead of sampling the randomness $\rho \xleftarrow{\$} \mathcal{R}$ uniformly at random from $\mathcal{R}$, $\mathcal{C}_1$ calls the `Rand` oracle in game $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ to obtain the randomness ρ. A query from $\mathcal{A}$ to the `HDSignO` oracle on input $(m, \mathbf{addr}_{\cdot,\cdot})$ is forwarded to the `RSign` oracle on input m and the randomness corresponding to $\mathbf{addr}_{\cdot,\cdot}$ of the $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{C}_1}_{\mathsf{RSig}}$ game. For a `HChildO` oracle query on input $(\mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$, $\mathcal{C}_1$ chooses a fresh key pair (independently of pk) $(\mathsf{sk}', \mathsf{pk}') \xleftarrow{\$} \mathsf{RSig.Gen}(\mathsf{par})$, assigns $(\mathsf{sk}_{\mathsf{h},i+1,t}, \mathsf{pk}_{\mathsf{h},i+1,t}) :- (\mathsf{sk}', \mathsf{pk}')$ and returns $\mathsf{pk}_{\mathsf{h},i+1,t}$. The `SKLeakO` oracle is then simulated by returning $\mathsf{sk}_{\mathsf{h},i+1,t}$ on input $\mathbf{addr}_{i+1,t}$. The simulation of the `HChildO` and `HDSignO` oracles cannot be distinguished by $\mathcal{A}$ from the oracles in $\boldsymbol{G}_3$ due to the rerandomizability of keys property of RSig. The only way in which $\mathcal{A}$ could detect the difference between $\boldsymbol{G}_3$ and the reduction provided by $\mathcal{C}_1$ would be if the following event occurs. $\mathcal{A}$ makes a random oracle query of the form $(\mathsf{sk}_{\mathsf{nh},i,s}, \cdot, \cdot)$ where $\mathsf{sk}_{\mathsf{nh},i,s}$ is either a non-hardened secret key that corresponds to a public key $\mathsf{pk}_{\mathsf{nh},i,s}$ previously output by the `NHChildO` oracle or $\mathsf{sk}_{\mathsf{nh},i,s}$ is the secret key corresponding to pk (if $\mathsf{sk}_{\mathsf{nh},i,s}$ belongs to a public key $\mathsf{pk}_{\mathsf{nh},i,s}$ can be efficiently checked via the function $\mathsf{ToPubKey}(\mathsf{sk}_{\mathsf{nh},i,s})$). By Claim C.1, this event happens with probability ϵ. However, when this event occurs, $\mathcal{C}_1$ learns the secret key $\mathsf{sk}_{\mathsf{nh},i,s}$ which it can use to compute the secret key sk of the $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$ game. This is due to the transitivity and invertible rerandomization property of RSig. $\mathcal{C}_1$ can then use sk to create a valid forgery in the $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{C}_1}_{\mathsf{RSig}}$ game. Therefore, we have that $\mathsf{Adv}^{\mathcal{C}_1}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} \geq \epsilon$. ■

It follows that $\Pr[\boldsymbol{G}_2^{\mathcal{A}}] \leq \Pr[\boldsymbol{G}_3^{\mathcal{A}}] + \epsilon$.

Game $\boldsymbol{G}_4$: This game works like the previous game with a modification to the `HChildO` oracle which works as follows. Let q_{sk} be the number of hardened nodes that $\mathcal{A}$ corrupts via the `SKLeakO` oracle. Upon $\mathcal{A}$ querying the `HChildO` oracle, with probability $\frac{1}{q_{\mathsf{sk}}+1}$, the address of this node is added to a list L. Let Bad define the event that a node corresponding to an address in L is corrupted.

Since the change in this game is only syntactical, $\mathcal{A}$'s winning probability is not affected by whether Bad occurs. It follows that $\Pr[\boldsymbol{G}_3^{\mathcal{A}}] = \Pr[\boldsymbol{G}_4^{\mathcal{A}}]$.

Game $\boldsymbol{G}_5$: This game works like the previous game with the only difference that $\boldsymbol{G}_5$ aborts in case event Bad occurs.

Lemma C.2 $\Pr[\boldsymbol{G}_4^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1] \cdot e$.

Proof. $\mathcal{A}$ can distinguish $\boldsymbol{G}_5$ from the previous game if the game aborts i.e., when the event Bad happens. This event happens for each $\texttt{SKLeakO}$ query, independently, with probability $\frac{1}{q_{\mathsf{sk}}+1}$. With probability $(1 - \frac{1}{q_{\mathsf{sk}}+1})$, a $\texttt{SKLeakO}$ oracle query does not lead to an abort. Hence, the overall probability with which the game does not abort on any $\texttt{SKLeakO}$ oracle query can be lower bounded by $(1 - \frac{1}{q_{\mathsf{sk}}+1})^{q_{\mathsf{sk}}} \geq e^{-1}$, i.e., Bad occurs with probability at most $1 - e^{-1}$. As we have argued that Bad occurs in $\boldsymbol{G}_4$ independently of the event $\boldsymbol{G}_4 = 1$, we have that $\Pr[\boldsymbol{G}_4^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1] \cdot 1/\Pr[\neg\mathsf{Bad}] \leq \Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1] \cdot e$.

■

Game $\boldsymbol{G}_6$: This game works like the previous game with a modification to the $\texttt{HChildO}$ oracle which works as follows. For the nodes that are chosen to be added to the list L, the game derives the public key of that node as a public key of a non-hardened node. The rest of the hardened nodes are generated as $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{RSig.Gen}(\mathsf{par})$ and assigned $(\mathsf{sk}_{\mathsf{h},i+1,t}, \mathsf{pk}_{\mathsf{h},i+1,t}) := (\mathsf{sk}, \mathsf{pk})$.

Lemma C.3 $\Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1]$.

Proof. $\mathcal{A}$ can distinguish $\boldsymbol{G}_6$ from the previous game if it corrupts a hardened node which is simulated as a non-hardened node i.e., one of the nodes in the list L. The only other way for $\mathcal{A}$ to distinguish these two games would be if $\mathcal{A}$ was able to query the random oracle on input the secret key of a non-hardened node as this would allow to recursively compute the secret key of the corresponding child hardened node. This case is, however, has already been excluded in $\boldsymbol{G_3}$. As explained in game $\boldsymbol{G}_5$, upon $\mathcal{A}$ making a corruption query for a node in list L, the game aborts. Therefore, the adversary cannot distinguish this game from the previous game.

■

By the transition from game $\boldsymbol{G}_0$ to game $\boldsymbol{G}_6$, we get that

$$\begin{aligned}
\mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal[RSig]}}} &= \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \\
&\leq \big(\Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] \cdot e\big) + \epsilon \\
\text{or, } \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] &\geq \frac{1}{e} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal[RSig]}}} - \frac{1}{e} \cdot \epsilon
\end{aligned}$$

Reduction to uf-cma-hrk security. Having shown that the transition from game $\mathbf{wufcma1}^{\mathcal{A}}_{\mathsf{HDWal[RSig]}}$ to the game $\boldsymbol{G}_6$ is indistinguishable, it remains to show that there exists a challenger $\mathcal{C}_2$ that simulates $\boldsymbol{G}_5$ and uses $\mathcal{A}$ to win its own game $\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}$. The challenger code is same as $\boldsymbol{G}_6$ with the following changes: (1) The sampling of $\rho \xleftarrow{\$} \mathcal{R}$ within the programming of H is replaced by a call to the oracle $\texttt{Rand}$ (2) pk is replaced by the challenge public key $\mathsf{pk}_{\mathcal{C}_2}$ from the underlying game $\mathbf{uf\text{-}cma\text{-}hrk1}^{\mathcal{C}_2}_{\mathsf{RSig}}$. Since the above changes are trivially indistinguishable to $\mathcal{A}$, we move on to analyze $\mathcal{C}_2$'s probability to win the $\mathbf{uf\text{-}cma\text{-}hrk}^{\mathcal{C}_2}_{\mathsf{RSig}}$ game using the forgery of $\mathcal{A}$. There are two possibilities for $\mathcal{A}$; either to output a forgery for a non-hardened node or for a hardened node. We analyze each case separately and show that for both cases our simulator can win its game with non-negligible probability.

- Adversary outputs a forgery for a non-hardened node: If the adversary provides a forgery for a non-hardened node, $\mathcal{C}_2$ can always use this forgery to win the $\mathbf{uf\text{-}cma\text{-}hrk}^{\mathcal{C}_2}_{\mathsf{RSig}}$ game. Therefore, the overall probability of $\mathcal{C}_2$ winning the game in case of $\mathcal{A}$ generating a forgery for a non-hardened node is:

$$\begin{aligned}
\mathsf{Adv}^{\mathcal{C}_2}_{\mathbf{uf\text{-}cma\text{-}hrk1}_{\mathsf{RSig}}} &\geq \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] \\
&\geq \frac{1}{e} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal[RSig]}}} - \frac{\epsilon}{e}
\end{aligned}$$

- Adversary outputs a forgery for a hardened node: We now compute the probability that the game aborts in case the adversary generates a forgery for a hardened node.

Let i^* be the index of the hardened node for which the adversary outputs a forgery. In this case $\mathcal{C}_2$ needs to abort if i^* was sampled randomly. Recall, the probability that i^* has been sampled at random is $1 - \frac{1}{q_{\mathsf{sk}}+1}$. Therefore, the overall probability of the simulator winning the game in case of $\mathcal{A}$ generating a forgery for a hardened node is:

$$\begin{aligned}\mathsf{Adv}^{\mathcal{C}_2}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} &\geq \Pr[\boldsymbol{G}^{\mathcal{A}}_6 = 1] \cdot \frac{1}{q_{\mathsf{sk}}+1} \\ &\geq \left(\frac{1}{e} \cdot \mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}} - \frac{\epsilon}{e}\right) \cdot \frac{1}{q_{\mathsf{sk}}+1}\end{aligned}$$

We can now compose a challenger $\mathcal{C}$ from the challengers $\mathcal{C}_1$ of Claim C.1 and $\mathcal{C}_2$, such that $\mathcal{C}$ uses adversary $\mathcal{A}$ to win in its game $\textbf{uf-cma-hrk1}^{\mathcal{C}}_{\mathsf{RSig}}$. $\mathcal{C}$ executes either of $\mathcal{C}_1$ and $\mathcal{C}_2$ with probability $\frac{1}{2}$. In order to compute $\mathcal{C}$'s advantage $\mathsf{Adv}^{\mathcal{C}}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}}$, we distinguish the following two cases:

- Case $\epsilon \geq \frac{1}{2}\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal}}}$: In this case, we have by claim C.1 that

  $$\mathsf{Adv}^{\mathcal{C}_1}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} \geq \epsilon \geq \frac{1}{2}\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal}}}.$$

  Therefore we can lower bound $\mathcal{C}$'s advantage by

  $$\mathsf{Adv}^{\mathcal{C}}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} \geq \frac{1}{2}\mathsf{Adv}^{\mathcal{C}_1}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} \geq \frac{\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal}}}}{4}.$$

- Case $\epsilon < \frac{1}{2}\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal}}}$: In this case, we can lower bound $\mathcal{C}_2$'s advantage by

  $$\begin{aligned}&\mathsf{Adv}^{\mathcal{C}_2}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} \\ &\geq \left(\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}} \cdot \frac{1}{e} - \frac{\epsilon}{e}\right) \cdot \frac{1}{q_{\mathsf{sk}}+1} \\ &\geq \left(\mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}} \cdot \frac{1}{e} - \frac{1}{2e} \cdot \mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}}\right) \cdot \frac{1}{q_{\mathsf{sk}}+1} \\ &= \frac{1}{2e(q_{sk}+1)} \cdot \mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}}\end{aligned}$$

Hence, $\mathcal{C}$'s overall advantage can be lower bounded by

$$\begin{aligned}\mathsf{Adv}^{\mathcal{C}}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} &\geq \min\left(\frac{1}{2}\mathsf{Adv}^{\mathcal{C}_1}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}}, \frac{1}{2}\mathsf{Adv}^{\mathcal{C}_2}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}}\right) \\ &\geq \frac{1}{4e(q_{sk}+1)} \cdot \mathsf{Adv}^{\mathcal{A}}_{\textbf{wufcma1}_{\mathsf{HDWal[RSig]}}}.\end{aligned}$$

■

D Impossibility of a tighter bound

In this section, we first recall the security notion of unforgeability under rerandomized keys for signature schemes with rerandomizable keys as introduced in [FKM+16]. This notion is stronger than the notion of unforgeability under *honestly* rerandomized keys in the sense that an adversary is not restricted to randomness chosen uniformly at random from the randomness space $\mathcal{R}$ for the key rerandomization. We recall the following security game:

Game $\textbf{uf-cma-rk}_{\mathsf{RSig}}$:

- **Setup Phase:** The challenger initializes the list $\mathsf{SigList} \leftarrow \{\epsilon\}$ and samples a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RSig.Gen}(\mathsf{par})$. Then the public key pk is sent to the adversary $\mathcal{A}$.

- **Online Phase:** $\mathcal{A}$ is given access to a signing oracle $\mathtt{RSign}$ which works as follows. On input a message m and a randomness ρ, derive a pair of keys rerandomized with the randomness ρ, as $\mathsf{sk}' \leftarrow \mathsf{RSig.SKDer}(\mathsf{sk}, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RSig.PKDer}(\mathsf{pk}, \rho)$. A signature is then derived on message m under the secret key sk' as $\sigma \leftarrow \mathsf{RSig.Sign}(\mathsf{sk}', m)$. The message m is stored in the $\mathsf{SigList}$ and eventually the signature σ is returned as the answer.

- **Output Phase:** Finally, the adversary $\mathcal{A}$ wins the game if it can provide a signature σ^* for a message m^* relative to randomness ρ^*, where the following holds: (1) the message m^* has not been queried before, i.e., $m^* \notin \mathsf{SigList}$, and (2) σ^* is a valid forgery, i.e., $\mathsf{RSig.Verify}(\mathsf{pk}^*, \sigma^*, m^*) = 1$, where $\mathsf{pk}^* \leftarrow \mathsf{RSig.PKDer}(\mathsf{pk}, \rho^*)$.

For an algorithm $\mathcal{A}$ we define $\mathcal{A}$'s advantage in game $\textbf{uf-cma-rk}_{\mathsf{RSig}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-rk}_{\mathsf{RSig}}} = \Pr[\textbf{uf-cma-rk}^{\mathcal{A}}_{\mathsf{RSig}} = 1]$.

We now show the proof of Theorem 5.4 as presented in Section 5. Concretely, we show that for any signature scheme with rerandomizable keys RSig that satisfies **uf-cma-rk** security and for any generic transformation from RSig to a hierarchical deterministic wallet scheme $\mathsf{HDWal}^{\mathsf{RSig}}$ there exists no reduction from $\textbf{uf-cma-rk}_{\mathsf{RSig}}$ to $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ that does not incur a loss polynomial in the number of $\mathtt{SKLeakO}$ oracle queries q_{sk}. In particular, this shows that the reduction in our proof of Theorem 5.3 is optimal and cannot be improved even assuming a generic transformation $\mathsf{HDWal}^{\mathsf{RSig}}$ from a **uf-cma-rk** secure signature scheme with rerandomizable keys. We show this result by assuming a reduction $\mathcal{R}$ that reduces $\textbf{uf-cma-rk}_{\mathsf{RSig}}$ to $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ and by providing a metareduction $\mathcal{M}$ that uses $\mathcal{R}$ to win its own $\textbf{uf-cma-rk}_{\mathsf{RSig}}$ game. We show that the advantage of $\mathcal{M}$ in game $\textbf{uf-cma-rk}_{\mathsf{RSig}}$ has a polynomial loss in the number of $\mathtt{SKLeakO}$ oracle queries q_{sk}. Our proof proceeds in a similar fashion as the proofs in [KK18, Theorem 2] and [Cor02, Theorem 4].

We now provide the full formal proof of Theorem 5.4.

Proof. We describe a metareduction $\mathcal{M}$ that plays in the game $\textbf{uf-cma-rk}^{\mathcal{M}}_{\mathsf{RSig}}$ and simulates the game $\textbf{uf-cma-rk}^{\mathcal{R}}_{\mathsf{RSig}}$ to $\mathcal{R}$. Additionally, $\mathcal{M}$ simulates an adversary in game $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ to $\mathcal{R}$. $\mathcal{M}$ receives a public key $\mathsf{pk}_{\mathcal{M}}$ from its challenger, and access to a signing oracle $\mathtt{RSign}$. The goal of $\mathcal{M}$ is to come up with a valid forgery in the $\textbf{uf-cma-rk}^{\mathcal{M}}_{\mathsf{RSig}}$ game. The metareduction proceeds as follows:

1. $\mathcal{M}$ runs the reduction $\mathcal{R}$ with public key $\mathsf{pk}_{\mathcal{M}}$ as input and simulates game $\textbf{uf-cma-rk}^{\mathcal{R}}_{\mathsf{RSig}}$ to $\mathcal{R}$ by simply forwarding $\mathcal{R}$'s queries to its own challenger. $\mathcal{R}$ sends a public key pk to $\mathcal{M}$ in the game $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$.

2. Assume that $\mathcal{M}$ in game $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ has made q queries to the $\mathtt{HChildO}$ oracle on input pairs $(\mathbf{addr}_{\cdot,\cdot}, \mathsf{e}^{\cdot,\cdot})$ and let $\mathcal{X}$ be a set consisting of the q addresses that $\mathcal{M}$ has queried the $\mathtt{HChildO}$ oracle on (for simplicity we write $\mathcal{X} = \{\mathbf{addr}_1, \cdots, \mathbf{addr}_q\}$). Let $q_{\mathsf{sk}} \leq \lfloor q/2 \rfloor$ be the number of addresses, for which $\mathcal{M}$ invokes the Secret Key Leakage oracle. $\mathcal{M}$ picks $i \xleftarrow{\$} \{1, \ldots, q_{\mathsf{sk}}\}$, chooses $\mathbf{addr}^* \xleftarrow{\$} \mathcal{X}$ and $(\mathbf{addr}_1, \cdots, \mathbf{addr}_{q_{\mathsf{sk}}}) \xleftarrow{\$} (\mathcal{X} \setminus \{\mathbf{addr}^*\})^{q_{\mathsf{sk}}}$. This defines the following two sequences:
$$\mathcal{X}_s := (\mathbf{addr}_1, \ldots, \mathbf{addr}_{i-1}, \mathbf{addr}^*)$$
$$\mathcal{X}'_s := (\mathbf{addr}_1, \cdots, \mathbf{addr}_{q_{\mathsf{sk}}})$$

3. $\mathcal{M}$ queries the $\mathtt{SKLeakO}$ oracle on addresses in the set $\mathcal{X}_s$ and receives the corresponding secret keys as answers from $\mathcal{R}$. In particular, since $\mathbf{addr}^* \in \mathcal{X}_s$, $\mathcal{M}$ knows the secret key sk^*.

4. $\mathcal{R}$ is then rewound to the initial state. Then $\mathcal{M}$, in game $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$, queries the $\mathtt{SKLeakO}$ oracle on addresses from the set $\mathcal{X}'_s$. Since $\mathbf{addr}^* \notin \mathcal{X}'_s$, $\mathcal{M}$ has not corrupted the node with $\mathbf{addr}^*$.

5. $\mathcal{M}$ now tosses a biased coin τ with probability $\epsilon_{\mathcal{A}}$ of outputting 1. If $\tau = 0$, $\mathcal{M}$ sends $\perp$ to $\mathcal{R}$ in the $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$ game. If $\tau = 1$, $\mathcal{M}$ samples a random message m, creates a signature σ on m with secret key sk^* and returns (σ, m) as a valid forgery. This execution is done in time $t_{\mathcal{A}}$ such that $\mathcal{M}$ correctly simulates an adversary in game $\textbf{wufcma1}_{\mathsf{HDWal}^{\mathsf{RSig}}}$.

6. Since $\mathcal{R}$ was rewound, sk^* was not revealed and (σ, m) constitutes a valid forgery. $\mathcal{R}$ derives a signature (σ', m') corresponding to challenge key $\mathsf{pk}_{\mathcal{M}}$ and returns it to $\mathcal{M}$. $\mathcal{M}$ can return (σ', m') to the $\textbf{uf-cma-rk}^{\mathcal{M}}_{\mathsf{RSig}}$ game.

Success probability of $\mathcal{M}$ We now analyze the probability with which $\mathcal{M}$ can win the $\mathbf{uf\text{-}cma\text{-}rk}^{\mathcal{M}}_{\mathsf{RSig}}$ game. Let $\mathcal{Q}$ be a set of sequences of addresses such that for any sequence $(\mathbf{addr}_1, \cdots, \mathbf{addr}_j) \in \mathcal{Q}$, the corresponding `SKLeakO` oracle queries are answered correctly by $\mathcal{R}$. Additionally, it holds that if $(\mathbf{addr}_1, \cdots, \mathbf{addr}_j) \in \mathcal{Q}$, then also $(\mathbf{addr}_1, \cdots, \mathbf{addr}_{j-1}) \in \mathcal{Q}$. Let us now consider a (possibly unbounded) real adversary $\mathcal{A}$ (i.e., $\mathcal{A}$ is not simulated by $\mathcal{M}$), who issues queries to the `SKLeakO` oracle on inputs $\mathbf{addr}_j \in \mathcal{X}'_s$ and eventually outputs a valid forgery (σ, m) with success probability $\epsilon_{\mathcal{A}}$. The view of $\mathcal{R}$ is exactly the same when interacting with the real adversary $\mathcal{A}$ or with the adversary who is simulated by $\mathcal{M}$ (which we denote by $\mathcal{A}_{\mathcal{M}}$) except if the following bad event occurs: $\mathcal{X}_s \notin \mathcal{Q}$ but $\mathcal{X}'_s \in \mathcal{Q}$. In this case, the reduction $\mathcal{R}$ did not answer all `SKLeakO` oracle queries correctly in the interaction with $\mathcal{A}_{\mathcal{M}}$ before the rewind but did so after the rewind. If this event occurs, the real adversary $\mathcal{A}$ would output a valid forgery, while the simulated adversary $\mathcal{A}_{\mathcal{M}}$ would not. Hence, the reduction $\mathcal{R}$ would be able to distinguish the real from the simulated execution.

Let $\mathcal{R}^{\mathcal{A}}$ and $\mathcal{R}^{\mathcal{A}_{\mathcal{M}}}$ denote the execution of the reduction w.r.t. the real and simulated adversary, respectively. The executions $\mathcal{R}^{\mathcal{A}}$ and $\mathcal{R}^{\mathcal{A}_{\mathcal{M}}}$ are identical, except if the following bad events occur in $\mathcal{R}^{\mathcal{A}_{\mathcal{M}}}$: $\mathcal{X}'_s \in \mathcal{Q}$ and $\mathcal{X}_s \notin \mathcal{Q}$ and $\tau = 1$. Therefore, we get:

$$\begin{aligned}
&|\Pr[(\sigma', m') \leftarrow \mathcal{R}^{\mathcal{A}_{\mathcal{M}}}(\mathsf{pk}_{\mathcal{M}}) \wedge \ (\sigma' \text{ is valid on } m')] \\
&- \Pr[(\sigma', m') \leftarrow \mathcal{R}^{\mathcal{A}}(\mathsf{pk}_{\mathcal{M}}) \wedge \ (\sigma' \text{ is valid on } m')]| \\
&\leq \epsilon_{\mathcal{A}} \cdot \Pr[\mathcal{X}'_s \in \mathcal{Q} \wedge \mathcal{X}_s \notin \mathcal{Q}].
\end{aligned}$$

We recall the following lemma due to Coron [Cor02].

Lemma D.1 *Let $\mathcal{Q}$ be a set of sequences of at most q_{sk} integers in $\mathcal{X}$, such that for any sequence $(\mathbf{addr}_1, \cdots, \mathbf{addr}_j) \in \mathcal{Q}$, we have $(\mathbf{addr}_1, \cdots, \mathbf{addr}_{j-1}) \in \mathcal{Q}$. Then:*

$$\Pr_{\substack{i \xleftarrow{\$} \{1, \cdots, q_{\mathsf{sk}}\} \\ (\mathbf{addr}_1, \cdots, \mathbf{addr}_{q_{\mathsf{sk}}}, \mathbf{addr}^*) \xleftarrow{\$} \mathcal{X}^{q_{\mathsf{sk}}+1}}} \left[\begin{array}{l} (\mathbf{addr}_1, \cdots, \mathbf{addr}_{q_{\mathsf{sk}}}) \in \mathcal{Q} \\ \wedge (\mathbf{addr}_1, \cdots, \mathbf{addr}_{i-1}, \mathbf{addr}^*) \notin \mathcal{Q} \end{array} \right] \leq \frac{\exp(-1)}{q_{\mathsf{sk}}}.$$

From lemma D.1, representing addresses as integers, we get that

$$\Pr[\mathcal{X}'_s \in \mathcal{Q} \wedge \mathcal{X}_s \notin \mathcal{Q}] \leq \frac{\exp(-1)}{q_{\mathsf{sk}}} \left(1 - \frac{q_{\mathsf{sk}}}{q}\right)^{-1}.$$

Note that the additional term $\left(1 - \frac{q_{\mathsf{sk}}}{q}\right)^{-1}$ comes from the fact that we chose all $\mathbf{addr}_i$ from the set $\mathcal{X} \setminus \{\mathbf{addr}^*\}$ instead of $\mathcal{X}$. Hence, we need to consider the probability that for all $\mathbf{addr}_i$ it holds that $\mathbf{addr}_i \neq \mathbf{addr}^*$.

From this, we obtain the success probability $\mathsf{Adv}^{\mathcal{M}}_{\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}}$ for $\mathcal{M}$ as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathcal{M}}_{\mathbf{uf\text{-}cma\text{-}rk}_{\mathsf{RSig}}} &= \Pr[(\sigma', m') \leftarrow \mathcal{R}^{\mathcal{A}_{\mathcal{M}}}(\mathsf{pk}_{\mathcal{M}}) \wedge \ (\sigma' \text{ is valid on } m')] \\
&\geq \Pr[(\sigma', m') \leftarrow \mathcal{R}^{\mathcal{A}}(\mathsf{pk}_{\mathcal{M}}) \wedge \ (\sigma' \text{ is valid on } m')] - \epsilon_{\mathcal{A}} \cdot \frac{\exp(-1)}{q_{\mathsf{sk}}} \left(1 - \frac{q_{\mathsf{sk}}}{q}\right)^{-1} \\
&\geq \Pr[(\sigma', m') \leftarrow \mathcal{R}^{\mathcal{A}}(\mathsf{pk}_{\mathcal{M}}) \wedge \ (\sigma' \text{ is valid on } m')] - \epsilon_{\mathcal{A}} \cdot \frac{2\exp(-1)}{q_{\mathsf{sk}}} \\
&\geq \epsilon_{\mathcal{R}}(\epsilon_{\mathcal{A}}) - \epsilon_{\mathcal{A}} \cdot \frac{2\exp(-1)}{q_{\mathsf{sk}}}
\end{aligned}$$

Note that, since $\mathcal{M}$ rewinds the reduction $\mathcal{R}$ once, the running time of $\mathcal{M}$ can be upper bounded by $t_{\mathcal{M}} \leq 2 \cdot t_{\mathcal{R}}(t_{\mathcal{A}})$.

■

E Discussion (contd.)

The following Theorem follows from Theorems 3.3 and 5.3.

Theorem E.1 *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be a hash function modeled as a random oracle. Let* $\mathsf{REC}[\mathsf{H}_1]$ *be the scheme as defined in Figure 7. Let* $\mathsf{HDWal}[\mathsf{RSig}]$ *be the construction as defined in Figure 5. We define* $\mathsf{HDWal}[\mathsf{REC}[\mathsf{H}_1]]$ *as the construction of* $\mathsf{HDWal}[\mathsf{RSig}]$*, instantiated with* $\mathsf{RSig} = \mathsf{REC}[\mathsf{H}_1]$*. Let* $\mathcal{A}$ *be an algorithm that plays in the game* $\mathbf{wufcma1}_{\mathsf{HDWal}[\mathsf{REC}[\mathsf{H}_1]]}$*, then there exists an algorithm* $\mathcal{C}$ *running in roughly the same time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathcal{C}}_{\mathbf{uf\text{-}cma1}_{\mathsf{EC}}[\mathsf{H}_0]} \geq \left(\frac{1}{4e(q_{sk}+1)} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal}[\mathsf{RSig}]}} - \frac{q_{\mathsf{H}_1}^2}{p} \right) \cdot \frac{1}{q},$$

where q_{H_1} *is the number of random oracle queries,* q *is the total number of* `HChildO` *and* `NHChildO` *oracle queries and* q_{sk} *is the number of queries to the* `SKLeakO` *oracle.*

The following Theorem follows from Theorem 5.3 and Theorem 5.1 from [DFL19].

Theorem E.2 *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions modeled as a random oracles. Let* $\mathsf{REC}'[\mathsf{H}_1]$ *be the scheme as defined in Figure 6. Let* $\mathsf{HDWal}[\mathsf{RSig}]$ *be the construction as defined in Figure 5. We define* $\mathsf{HDWal}[\mathsf{REC}'[\mathsf{H}_1]]$ *as the construction of* $\mathsf{HDWal}[\mathsf{RSig}]$*, instantiated with* $\mathsf{RSig} = \mathsf{REC}'[\mathsf{H}_1]$*. Let* $\mathcal{A}$ *be an algorithm that plays in the game* $\mathbf{wufcma1}_{\mathsf{HDWal}[\mathsf{REC}'[\mathsf{H}_1]]}$*, then there exists an algorithm* $\mathcal{C}$ *running in roughly the same time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathcal{C}}_{\mathbf{uf\text{-}cma1}_{\mathsf{EC}}[\mathsf{H}_0]} \geq \frac{1}{4e(q_{sk}+1)} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathbf{wufcma1}_{\mathsf{HDWal}[\mathsf{RSig}]}} - \frac{3{q_{\mathsf{H}_1}}^2}{p},$$

where q_{H_1} *is the number of random oracle queries and* q_{sk} *is the number of queries to the* `SKLeakO` *oracle.*

Lemma E.3 *Consider the algorithm* $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]_{\mathsf{EC}}$ *in Figure 8. Suppose that:*

- $\omega = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$,
- $X_0, X_1 \in \mathbb{E}$ *s.t.* $X_0 = x_0 \cdot G$ *and* $X_1 = \omega \cdot X_0$,
- $\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(X_1, \sigma_1, m_1) = 1$,
- $\sigma_0 \leftarrow \mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]_{\mathsf{EC}}(m_0, m_1, \sigma_1, \omega, X_0, X_1)$.

Then $\mathsf{EC}[\mathsf{H}_0].\mathsf{Verify}(X_0, \sigma_0, m_0) = 1$.

Theorem E.4 *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be a hash function modeled as a random oracle. Let* $\mathcal{A}$ *be an algorithm that plays in the game* $\mathbf{uf\text{-}cma\text{-}hrk}_{\mathsf{MREC}[\mathsf{H}_1]}$*, then there exists an algorithm* $\mathcal{C}$ *running in roughly the same time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathcal{C}}_{\mathbf{uf\text{-}cma}_{\mathsf{EC}}[\mathsf{H}_0]} \geq \mathsf{Adv}^{\mathcal{A}}_{\mathbf{uf\text{-}cma\text{-}hrk}_{\mathsf{MREC}[\mathsf{H}_1]}} - \frac{3q^2}{p}$$

Proof. Consider an adversary $\mathcal{A}$ playing in Game $\mathbf{uf\text{-}cma\text{-}hrk}_{\mathsf{MREC}[\mathsf{H}_1]}$. As such $\mathcal{A}$ is granted access to the oracles `Rand`, `RSign`, and the random oracle $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$. In the following, we use that $2^\kappa \leq p$. We prove the statement via a sequence of games. Each game $\boldsymbol{G}_{i(i>0)}$ is presented in Figure 10 via the description of the oracles that are modified with respect to the previous game $\boldsymbol{G}_{i-1}$. The exact differences of game $\boldsymbol{G}_i$ to game $\boldsymbol{G}_{i-1}$ are highlighted in the form of boxed pseudocode. Moreover, we denote by $E_{i-1,i}$ a difference event, where the indices of the event correspond to games $\boldsymbol{G}_{i-1}, \boldsymbol{G}_i$ that are affected by the event.

Game $\boldsymbol{G}_0$: This game is equivalent to the original $\mathbf{uf\text{-}cma\text{-}hrk}_{\mathsf{MREC}[\mathsf{H}_1]}$ game. In particular, a key pair $(\mathsf{sk}, \mathsf{pk})$ is sampled as $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{MREC}[\mathsf{H}_1].\mathsf{Gen}(\mathsf{par})$. The adversary $\mathcal{A}$ is given pk as the challenge public key and oracle access to `Rand`, `RSign` and random oracle H_1. $\mathcal{A}$ can query `Rand` to receive a randomness ρ and make a follow-up query to $\texttt{RSign}$ to receive a signature on message m with respect to the rerandomized key $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$. In particular, $\mathcal{A}$ is allowed to query $\texttt{RSign}$ on every input pair (m, ρ) at most once. Additionally, $\mathcal{A}$ can make direct queries to the random oracle H_1. The game internally maintains a random oracle H_0 in a straightforward manner, by storing a list H_0 of query/response pairs. Eventually, in order to win the game, $\mathcal{A}$ has to come up with a valid forgery σ^* on a new message m^* with respect to a randomness ρ^*. Since $\boldsymbol{G_0}$ proceeds as $\textbf{uf-cma-hrk}_{\mathsf{MREC}[\mathsf{H}_1]}$ we have that $\Pr[\boldsymbol{G_0} = 1] = \Pr[\textbf{uf-cma-hrk}_{\mathsf{MREC}[\mathsf{H}_1]} = 1] = \mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk}_{\mathsf{MREC}[\mathsf{H}_1]}}$.

Game $\boldsymbol{G}_1$: In $\boldsymbol{G}_1$, the way that random oracle queries to H_1 from $\mathcal{A}$ are answered, is internally modified as follows. To answer queries to H_1, $\boldsymbol{G}_1$ internally keeps two lists H_1 and H_1' which it programs throughout its interaction with $\mathcal{A}$. Depending on whether a queried message m contains as part of its prefix a public key pk', it programs $H_1[m]$ and $H_1'[m]$ in two different possible ways. Note that pk' is the result of rerandomizing pk as $\mathsf{pk}' = \mathsf{pk} \cdot \rho$, where $\rho \leftarrow \texttt{Rand}(\rho \in \mathsf{RList})$ is a previous answer to an oracle query $\texttt{Rand}$. We now analyze the three types of queries to H_1 that can occur.

- $\mathsf{H}_1[m] \neq \perp$: In this case, $\boldsymbol{G}_1$ returns $H_1[m]$.
- $\mathsf{H}_1[m] = \perp$ and m is of the form $m = (\mathsf{pk}', \cdot)$, s.t. $\mathsf{pk}' = \mathsf{pk} \cdot \rho$ for some $\rho \in \mathsf{RList}$: In this case, $\boldsymbol{G}_1$ computes $h \leftarrow \mathsf{H}_0(ctr)$, where $ctr \xleftarrow{\$} \{0,1\}^\kappa$. Consequently, $\boldsymbol{G}_1$ sets $H_1[m] \leftarrow \rho \cdot h \mod p$ and $H_1'[m] \leftarrow ctr$. It returns $H_1[m]$.
- Otherwise, $\boldsymbol{G}_1$ samples $h \xleftarrow{\$} \mathbb{Z}_p$ and sets the values $H_1[m] \leftarrow h$, $H_1'[m] \leftarrow \epsilon$. It then returns $H_1[m]$.

It is easy to see that all answers for queries to H_1 that $\boldsymbol{G}_1$ returns are uniformly distributed from $\mathcal{A}$'s perspective. This follows from the uniformity of output h computed via random oracle H_0. Therefore, $\boldsymbol{G}_1$ behaves exactly as $\boldsymbol{G}_0$.

Game $\boldsymbol{G}_2$: In $\boldsymbol{G}_2$, the way in which queries to $\texttt{Rand}$ are answered, is internally modified as follows. When $\mathcal{A}$ asks a query of the form $\texttt{Rand}$, the game aborts if there exists a message of the form $m = (\mathsf{pk}', \cdot)$ for which $H_1'[m]$ evaluates to ϵ and where pk' is the (rerandomized) key that corresponds to the return value ρ of $\texttt{Rand}$, i.e., $\mathsf{pk}' = \mathsf{pk} \cdot \rho$. The following claim bounds the probability of such an abort scenario.

Claim E.5 Let $E_{1,2}$ denote the event that $\boldsymbol{G}_2$ aborts during a $\texttt{Rand}$ query, for which $H_1'[m]$ evaluates to ϵ, where $m = (\mathsf{pk}', \cdot)$. Then $\Pr[E_{1,2}] \leq \frac{q^2}{p}$.

Proof. During any particular call to the oracle $\texttt{Rand}$, this event can only occur if $\mathcal{A}$ has already made a query of the form $\mathsf{H}_1(m)$, where $m = (\mathsf{pk}', \cdot)$ (prior to the oracle $\texttt{Rand}$ returning the value ρ for this query). Since RList contains at most q values at any point during the game, any of them coincide with the (uniformly chosen) value ρ with probability at most $\frac{q}{p}$. Since keys are uniquely rerandomizable, a query of the form $\mathsf{H}_1(m)$ thus also has probability at most $\frac{q}{p}$ of having been made prior to this particular call to $\texttt{Rand}$. Since there at most q queries to $\texttt{Rand}$, it follows that $\Pr[E_{1,2}] \leq \frac{q^2}{p}$. ■

Since the games $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ are equivalent unless the event $\Pr[E_{1,2}]$ occurs, $\Pr[\boldsymbol{G}_0 = 1] \leq \Pr[\boldsymbol{G}_1] + \frac{q^2}{p}$

Game $\boldsymbol{G}_3$: In $\boldsymbol{G}_3$, the way that signing queries from $\mathcal{A}$ are answered, is again internally modified as follows. When $\mathcal{A}$ makes a query of the form $\texttt{RSign}(m, \rho)$, $\boldsymbol{G}_3$ first checks whether $\rho \in \mathsf{RList}$ and if not, returns $\perp$. Otherwise, it computes $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$, and sets $\mathsf{pm} \leftarrow (\mathsf{pk}', m)$. If $H_1[\hat{m}] = \perp$, it internally queries H_1 on input message pm. This means it queries $h \leftarrow \mathsf{H}(ctr)$, where $ctr \xleftarrow{\$} \{0,1\}^\kappa$. $\boldsymbol{G}_3$ internally sets $H_1[\mathsf{pm}] \leftarrow \rho \cdot h \mod p$ and stores $H_1'[\mathsf{pm}] \leftarrow ctr$. After making the query to H_1, $\boldsymbol{G}_3$ fetches $m' \leftarrow H_1'[\mathsf{pm}]$, where m' was set to ctr during H_1 query. Since sk is known to the game, it can now compute the signature σ' as $\sigma' \xleftarrow{\$} \mathsf{EC}[H_0].\mathsf{Sign}(\mathsf{sk}, m')$. Finally, it computes and returns the signature σ as $\sigma \leftarrow \mathsf{Trf}[H_0, \mathsf{H}_1]_{\mathsf{EC}}(\mathsf{pm}, m', \sigma', \rho^{-1}, \mathsf{pk}', \mathsf{pk})$, where $\mathsf{pk} = \mathsf{pk}' \cdot \rho^{-1}$.

Claim E.6 $\Pr[\boldsymbol{G}_2 = 1] = \Pr[\boldsymbol{G}_3 = 1]$

Proof. We argue that in both games, the answers to signing queries are identically distributed. To this end, we analyze how $\boldsymbol{G}_3$ replies to a query of the form $\texttt{RSign}(m, \rho)$. $\boldsymbol{G}_3$ derives signature σ on input (m, ρ) as $\sigma \leftarrow \mathsf{Trf}[H_0, \mathsf{H}_1]_{\mathsf{EC}}(\mathsf{pm}, m', \sigma', \rho^{-1}, \mathsf{pk}', \mathsf{pk})$, where $m' = H_1'[\mathsf{pm}]$, $\mathsf{pk} = \mathsf{pk}' \cdot \rho^{-1}$, $\mathsf{EC}[H_0].\mathsf{Verify}(\mathsf{pk}, \sigma', m') = 1$, and $\frac{H_0[m']}{H_1[\mathsf{pm}]} = \frac{h'}{H_1[\mathsf{pm}]} = \frac{h'}{\rho \cdot h'} = \rho^{-1} \mod p$. It follows from Lemma E.3 that σ constitutes a correct signature on message pm and under public key pk' relative to $\mathsf{EC}[H_0].\mathsf{Verify}$. It follows immediately that the signature σ constitutes a valid signature relative to $\mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}$. This concludes the proof. ■

GAME $\boldsymbol{G}_4$: $\boldsymbol{G}_4$ behaves identically to $\boldsymbol{G}_3$ except for the following modification in the main procedure: Upon receiving a forgery of the form (m^*, σ^*, ρ^*) from $\mathcal{A}$, it sets $\mathsf{pm}^* \leftarrow (\mathsf{pk}^*, m^*)$ and aborts if $H_1'[\mathsf{pm}^*] = \epsilon$.

Claim E.7 Let $E_{3,4}$ be the event that $\boldsymbol{G}_4$ aborts if $H_1'[\mathsf{pm}^*] = \epsilon$, where $\mathsf{pm}^* = (\mathsf{pk}^*, m^*)$. Then $\Pr[E_{3,4}] \leq \frac{q^2}{p}$.

Proof. The only way this event can happen, is if $\mathcal{A}$ manages to make a query of the form $\mathsf{H}_1(\mathsf{pm}^*)$ before querying `Rand` to obtain the corresponding value of ρ^*. The proof of this claim follows in a similar way as the corresponding proof in claim E.5. ■

Since the games $\boldsymbol{G}_3$, $\boldsymbol{G}_4$ are equivalent unless event $E_{3,4}$ occurs, $\Pr[\boldsymbol{G}_3 = 1] \leq \Pr[\boldsymbol{G}_4 = 1] + \frac{q^2}{p}$
Reduction to UF-CMA security. We describe an algorithm $\mathcal{C}$ that plays in the $\textbf{uf-cma}_{\mathsf{EC}[\mathsf{H}_0]}$ game and simulates game $\boldsymbol{G}_4$ to $\mathcal{A}$. Instead of sampling its own key pair as is done in $\boldsymbol{G}_4$, $\mathcal{C}$ obtains as input a public key $\mathsf{pk}_{\mathcal{C}}$ from the $\textbf{uf-cma}_{\mathsf{EC}[\mathsf{H}_0]}$ game and is given access to the signing oracle `Sign` to obtain signatures under $\mathsf{pk}_{\mathcal{C}}$ under messages of its choice. Furthermore, $\mathcal{C}$ has access to the random oracle H_0 by which it replaces the list H_0. $\mathcal{C}$ runs $\mathcal{A}$ on input $\mathsf{pk}_{\mathcal{C}}$.
Simulation of Randomness Queries. Queries to `Rand` from $\mathcal{A}$ do not require knowledge of the secret key corresponding to $\mathsf{pk}_{\mathcal{C}}$ and hence are straight forward to simulate.
Simulation of Random Oracle Queries. $\mathcal{C}$'s simulation of random oracle queries coincides with the above programming strategy that is already internally present in $\boldsymbol{G}_4$.
Simulation of Signing Queries. Recall that in $\boldsymbol{G}_4$, queries of the form `RSign` (m, ρ) internally prompt the computation of signature $\sigma' = \mathsf{EC}[\mathsf{H}_0].\mathsf{Sign}(\mathsf{sk}_{\mathcal{C}}, m')$, where $m' \leftarrow ctr$. Since $\mathcal{C}$ does not know $sk_{\mathcal{C}}$, it needs to compute σ' via a call to its signing oracle, i.e., as $\sigma' \leftarrow \texttt{Sign}(m')$. Other than that $\mathcal{C}$ simulates such a query exactly as internally done for $\boldsymbol{G}_4$.

EXTRACTING THE FORGERY. When the tuple (m^*, σ^*, ρ^*) is returned as an answer from $\mathcal{A}$, $\mathcal{C}$ checks whether it constitutes a valid forgery, and aborts otherwise (note that in this case, $\boldsymbol{G}_4$ would return 0, so $\mathcal{C}$ can safely abort). In case $\mathcal{C}$ does not abort, it computes $\mathsf{pk}^* = \mathsf{pk}_{\mathcal{C}} \cdot \rho^*$, where pk^* is the public key under which $\mathcal{A}$'s forgery is valid. $\mathcal{C}$ computes $\mathsf{pm}^* \leftarrow (\mathsf{pk}^*, m^*)$ and if $H_1'[\mathsf{pm}^*] = \epsilon$, it aborts. Otherwise, $\mathcal{C}$ fetches $m' \leftarrow H_1'[\mathsf{pm}^*]$ and computes

$$\sigma' \leftarrow \mathsf{Trf}[\mathsf{H}_1, \mathsf{H}_0]_{\mathsf{ECDSA}}(m', \mathsf{pm}^*, \sigma^*, \rho^*, \mathsf{pk}_{\mathcal{C}}, \mathsf{pk}^*).$$

Since $H_1[\mathsf{pm}^*] = \mathsf{H}_0(H_1'[\mathsf{pm}^*]) \cdot \rho^* = \mathsf{H}(m') \cdot \rho^*$, we have that $\frac{H_1[\mathsf{pm}^*]}{\mathsf{H}_0(m')} = \frac{\mathsf{H}_0(m') \cdot \rho^*}{\mathsf{H}_0(m')} = \rho^*$. Together with $\mathsf{pk}^* = \mathsf{pk}_{\mathcal{C}} \cdot \rho^*$ and $\mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}^*, \sigma^*, \mathsf{pm}^*) = 1$, Lemma E.3 implies that

$$\mathsf{EC}[\mathsf{H}_0].\mathsf{Verify}(\mathsf{pk}_{\mathcal{C}}, \sigma', m') = 1.$$

Claim E.8 (m', σ') constitutes a valid forgery in $\textbf{uf-cma}_{\mathsf{EC}[\mathsf{H}_0]}$ with probability $1 - q^2/p$.

Proof. We have to show that the query $\texttt{Sign}(m')$ was not made by $\mathcal{C}$ during its simulation and hence (m', σ') is a valid forgery in $\textbf{uf-cma}_{\mathsf{EC}[\mathsf{H}_0]}$. Note that $\mathcal{A}$ has not made a query of the form `RSign` (m^*, ρ^*) throughout the simulation. If it had, (m^*, σ^*, ρ^*) would not constitute a valid forgery in $\boldsymbol{G}_4$ and the simulation would have aborted at this point. This implies that $\mathcal{C}$ never had to simulate a query $\texttt{RSign}(m^*, \rho^*)$ to $\mathcal{A}$ which entailed a H_1 query on message $\mathsf{pm}^* \leftarrow (\mathsf{pk}^*, m^*)$. Hence, m' associated with query $\mathsf{H}_1(\mathsf{pm}^*)$ was not queried by $\mathcal{C}$ to the oracle `Sign` in any query of the form `RSign` (m, ρ) with $m \neq m^*$ unless there exist (any) two values m_1, m_2 s.t. $H_1'[m_1] = H_1'[m_2] \neq \perp$. It is easy to see that this happens with probability at most q^2/p during $\mathcal{C}$'s simulation, since all values that $\mathcal{C}$ queries to the oracle `Sign` are sampled independently and uniformly at random from $\{0,1\}^\kappa$. ■

From claims E.5-E.7, we have $\Pr[\boldsymbol{G}_0 = 1] \leq \Pr[\boldsymbol{G}_4] + \frac{3q^2}{p}$ Since $\mathcal{C}$ provides a perfect simulation of $\boldsymbol{G}_4$ to $\mathcal{A}$ up to an error of q^2/p, as shown in the previous claim, we obtain

$$\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk},\mathsf{MREC}[\mathsf{H}_1]} \leq \mathsf{Adv}^{\mathcal{A}}_{\boldsymbol{G}_4} + \frac{3q^2}{p} \leq \mathsf{Adv}^{\mathcal{C}}_{\textbf{uf-cma},\mathsf{EC}[\mathsf{H}_0]} + \frac{3q^2}{p},$$

which implies the theorem.

■

Theorem 5.3 can be combined with Theorem E.4 in a similar manner as Theorem E.2.

```
Algorithm $\mathsf{HDWal[RSig].Setup(par)}$
00 $\mathsf{ch}_{0,0} \xleftarrow{\$} \{0,1\}^\kappa$
01 $(\mathsf{msk}_{0,0}, \mathsf{mpk}_{0,0}) \xleftarrow{\$} \mathsf{RSig.Gen(par)}$
02 Return $(\mathsf{msk}_{0,0}, \mathsf{mpk}_{0,0}, \mathsf{ch}_{0,0})$

Algorithm $\mathsf{HDWal[RSig].Sign}(\mathsf{sk}_{i,s}, m)$
00 $\sigma \leftarrow \mathsf{RSig.Sign}(\mathsf{sk}_{i,s}, m)$
01 Return $\sigma$

Algorithm $\mathsf{HDWal[RSig].Verify}(\mathsf{pk}_{i,s}, \sigma, m)$
00 $0/1 \leftarrow \mathsf{RSig.Verify}(\mathsf{pk}_{i,s}, \sigma, m)$
01 Return $0/1$

Algorithm $\mathsf{HDWal[RSig].SKDer_H}(\mathsf{sk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
00 $(\omega, \mathsf{ch}_{i+1,t}) \leftarrow \mathsf{H}(\mathsf{sk}_{i,s}, \mathsf{ch}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
01 $\mathsf{sk}_{i+1,t} \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}_{i,s}; \omega)$
02 $\mathbf{addr}_{i+1,t} \leftarrow \mathbf{addr}_{i,s} \parallel \mathsf{e}_{i+1}^{s,t}$
03 Return $(\mathsf{sk}_{i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t})$

Algorithm $\mathsf{HDWal[RSig].SKDer_{NH}}(\mathsf{sk}_{i,s}, \mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
00 $(\omega, \mathsf{ch}_{i+1,t}) \leftarrow \mathsf{H}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
01 $\mathsf{sk}_{i+1,t} \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}_{i,s}; \omega)$
02 $\mathbf{addr}_{i+1,t} \leftarrow \mathbf{addr}_{i,s} \parallel \mathsf{e}_{i+1}^{s,t}$
03 Return $(\mathsf{sk}_{i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t})$

Algorithm $\mathsf{HDWal[RSig].PKDer_H}(\mathsf{sk}_{i,s}, \mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
00 $(\omega, \mathsf{ch}_{i+1,t}) \leftarrow \mathsf{H}(\mathsf{sk}_{i,s}, \mathsf{ch}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
01 $\mathsf{pk}_{i+1,t} \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}_{i,s}; \omega)$
02 $\mathbf{addr}_{i+1,t} \leftarrow \mathbf{addr}_{i,s} \parallel \mathsf{e}_{i+1}^{s,t}$
03 Return $(\mathsf{pk}_{i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t})$

Algorithm $\mathsf{HDWal[RSig].PKDer_{NH}}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathbf{addr}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
00 $(\omega, \mathsf{ch}_{i+1,t}) \leftarrow \mathsf{H}(\mathsf{pk}_{i,s}, \mathsf{ch}_{i,s}, \mathsf{e}_{i+1}^{s,t})$
01 $\mathsf{pk}_{i+1,t} \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}_{i,s}; \omega)$
02 $\mathbf{addr}_{i+1,t} \leftarrow \mathbf{addr}_{i,s} \parallel \mathsf{e}_{i+1}^{s,t}$
03 Return $(\mathsf{pk}_{i+1,t}, \mathsf{ch}_{i+1,t}, \mathbf{addr}_{i+1,t})$
```

Figure 5: Generic construction of a hierarchical deterministic wallet scheme HDWal[RSig] from a signature with perfectly rerandomizable keys RSig. HDWal[RSig] is defined w.r.t. an address structure $(\mathcal{T}, \mathbf{Addr})$, where $\mathcal{T} = (h, n_{0,0}, \mathcal{N}, \mathcal{E})$, such that $\mathbf{addr}_{i,s} \in \mathbf{Addr}$ and $\mathsf{e}_i^{s,t} \in \mathcal{E}$ for $0 \leq i \leq h$ and $1 \leq s, t \leq |\mathcal{N}|$. We denote by $(\mathsf{pk}_{i,s}, \mathsf{sk}_{i,s})$ and $\mathsf{ch}_{i,s}$ the public/secret key pair and chaincode of the node with address $\mathbf{addr}_{i,s}$. We denote by H a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathcal{R} \times \{0,1\}^\kappa$.

```
Algorithm $\mathsf{REC}'[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}, m)$
00 $\psi \xleftarrow{\$} \{0,1\}^\kappa$
01 $\hat{m} \leftarrow (\mathsf{pk}, \psi, m)$
02 $\sigma' \leftarrow \mathsf{EC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{sk}, \hat{m})$
03 Return $\sigma = (\psi, \sigma')$

Algorithm
$\mathsf{REC}'[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}, \sigma, m)$
04 $(\psi, \sigma') \leftarrow \sigma$
05 $\hat{m} \leftarrow (\mathsf{pk}, \psi, m)$
06 Return $\mathsf{EC}[\mathsf{H}_1].(\mathsf{pk}, \sigma', \hat{m})$

Algorithm
$\mathsf{REC}'[\mathsf{H}_1].\mathsf{RandSK}\,(\mathsf{sk}; \rho)$
00 $\mathsf{sk}' \leftarrow \mathsf{sk} \cdot \rho \bmod p$
01 Return $\mathsf{sk}'$

Algorithm
$\mathsf{REC}'[\mathsf{H}_1].\mathsf{RandPK}\,(\mathsf{pk}; \rho)$
02 $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$
03 Return $\mathsf{pk}'$
```

Figure 6: Salted and key-prefixed version of the ECDSA signature scheme with perfectly rerandomizable keys $\mathsf{REC}'[\mathsf{H}_1] := (\mathsf{REC}'[\mathsf{H}_1].\mathsf{Gen} = \mathsf{EC}[\mathsf{H}_1].\mathsf{Gen}$, $\mathsf{REC}'[\mathsf{H}_1].\mathsf{Sign}$, $\mathsf{REC}'[\mathsf{H}_1].\mathsf{Verify}$, $\mathsf{REC}'[\mathsf{H}_1].\mathsf{RandSK}$, $\mathsf{REC}'[\mathsf{H}_1].\mathsf{RandPK})$ from the ECDSA signature scheme $\mathsf{EC}[\mathsf{H}_1]$. $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ denotes a hash function.

```
Algorithm
$\mathsf{MREC}[\mathsf{H}_1].\mathsf{Sign}\,(\mathsf{sk}, m)$
00 $\mathsf{pm} \leftarrow (\mathsf{pk}, m)$
01 $\sigma \leftarrow \mathsf{EC}[\mathsf{H}_1].\mathsf{Sign}\,(\mathsf{sk}, \mathsf{pm})$
02 Return $\sigma$

Algorithm
$\mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}\,(\mathsf{pk}, \sigma, m)$
03 $\mathsf{pm} \leftarrow (\mathsf{pk}, m)$
04 Return
$\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}\,(\mathsf{pk}, \sigma', \mathsf{pm})$

Algorithm
$\mathsf{MREC}[\mathsf{H}_1].\mathsf{RandSK}\,(\mathsf{sk}; \rho)$
00 $\mathsf{sk}' \leftarrow \mathsf{sk} \cdot \rho \bmod p$
01 Return $\mathsf{sk}'$

Algorithm
$\mathsf{MREC}[\mathsf{H}_1].\mathsf{RandPK}\,(\mathsf{pk}; \rho)$
02 $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$
03 Return $\mathsf{pk}'$
```

Figure 7: Salt-free and key-prefixed version of the ECDSA signature scheme with perfectly rerandomizable keys $\mathsf{MREC}[\mathsf{H}_1] := (\mathsf{MREC}[\mathsf{H}_1].\mathsf{Gen} = \mathsf{EC}[\mathsf{H}_1].\mathsf{Gen}$, $\mathsf{MREC}[\mathsf{H}_1].\mathsf{Sign}$, $\mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}$, $\mathsf{MREC}[\mathsf{H}_1].\mathsf{RandSK}$, $\mathsf{MREC}[\mathsf{H}_1].\mathsf{RandPK})$ from the ECDSA signature scheme $\mathsf{EC}[\mathsf{H}_1]$. $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ denotes a hash function.

```
$\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]_{\mathsf{EC}}(m_0, m_1, \sigma_1, \omega, X_0, X_1)$
00 $z_0 \leftarrow \mathsf{H}_0(m_0)$
01 $z_1 \leftarrow \mathsf{H}_1(m_1)$
02 If $(\mathsf{EC}[\mathsf{H}_1].\mathsf{Verify}(\sigma_1, m_1, X_1) = 0) \vee \left(\omega \neq \frac{z_1}{z_0} \vee X_1 \neq X_0 \cdot \omega\right)$ :
03     Return $\perp$
04 $(r, s_1) \leftarrow \sigma_1$
05 $s_0 \leftarrow \frac{s_1}{\omega} \bmod p$
06 $\sigma_0 \leftarrow (r, s_0)$
07 Return $\sigma_0$
```

Figure 8: Figure shows the $\mathsf{Trf}_{\mathsf{ECDSA}}$ algorithm for hash functions $\mathsf{H}_0, \mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$.

```
Game $G_0$
00 $\mathsf{RList} \leftarrow \{\epsilon\}$
01 $\mathsf{bad} \leftarrow \mathsf{false}$
02 $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{MREC}[\mathsf{H}_1].\mathsf{Gen}(\mathsf{par})$
03 $(m^*, \sigma^*, \rho^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{H}_1, \mathtt{Rand}, \mathtt{RSign}}(\mathsf{pk})$
04 $\mathsf{pk}^* \leftarrow \mathsf{pk} \cdot \rho^*$
05 If $\mathsf{pm}^* \in \mathsf{SigList}$: $\mathsf{bad} \leftarrow \mathsf{true}$
06 If $\rho^* \notin \mathsf{RList}$: $\mathsf{bad} \leftarrow \mathsf{true}$
07 $b \leftarrow \mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}^*, \sigma^*, m^*)$
08 Return $b \wedge \neg\mathsf{bad}$

Oracle $\mathtt{Rand}$
09 $\rho \xleftarrow{\$} \mathcal{R}$
10 $\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
11 Return $\rho$
```

```
Oracle $\mathtt{RSign}(m, \rho)$
12 If $\rho \notin \mathsf{RList}$: Return $\perp$
13 $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho \mod p$
14 $\mathsf{sk}' \leftarrow \mathsf{sk} \cdot \rho \mod p$
15 $\mathsf{pm} \leftarrow (\mathsf{pk}', m)$
16 $\sigma \leftarrow \mathsf{MREC}[\mathsf{H}_1].\mathsf{Sign}(\mathsf{pm}, \mathsf{sk}')$
17 $\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{\mathsf{pm}\}$
18 Return $\sigma$

Oracle $\mathsf{H}_1(m)$
19 If $H_1[m] \neq \perp$
20     Return $H_1[m]$
21 $H_1[m] \xleftarrow{\$} \mathbb{Z}_p$
22 Return $H_1[m]$

$H_0[m]$
23 If $H_0[m] \neq \perp$
24     Return $H_0[m]$
25 $H_0[m] \xleftarrow{\$} \mathbb{Z}_p$
26 Return $H_0[m]$
```

Figure 9: Game $G_0 = \textbf{uf-cma-hrk}_{\mathsf{MREC}[\mathsf{H}_0]}$ with adversary $\mathcal{C}$.

```
Oracle $\mathsf{H}_1(m)$ in $G_1$
00 If $H_1[m] \neq \perp$
01     Return $H_1[m]$
02 Parse $m$ as $(\mathsf{pk}', \cdot)$
03 If $\exists \rho \in \mathsf{RList} : \mathsf{pk}' = \mathsf{pk} \cdot \rho$
04     $ctr \leftarrow \{0,1\}^\kappa$
05     $h \leftarrow H_0[ctr]$
06     $H_1[m] \leftarrow \rho \cdot h \mod p$
07     $H_1'[m] \leftarrow ctr$
08 Else
09     $h \xleftarrow{\$} \mathbb{Z}_p$
10     $H_1[m] \leftarrow h$
11     $H_1'[m] \leftarrow \epsilon$
12 Return $H_1[m]$

Oracle $\mathtt{Rand}$ in $G_2$
13 $\rho \xleftarrow{\$} \mathcal{R}$
14 $\boxed{\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho}$
15 $\boxed{\forall m = (\mathsf{pk}', \cdot):}$
16 $\boxed{\text{If } H_1'[m] = \epsilon\text{: Abort}}$
17 $\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
18 Return $\rho$
```

```
Oracle $\mathtt{RSign}(m, \rho)$ in $G_3$
19 If $\rho \notin \mathsf{RList}$: Return $\perp$
20 $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot \rho$
21 $\mathsf{pm} \leftarrow (\mathsf{pk}', m)$
22 $\boxed{\text{If } H_1'[\mathsf{pm}] = \perp}$
23     $\boxed{\text{Query } \mathsf{H}_1(\mathsf{pm})}$
24 $\boxed{m' \leftarrow H_1'[\mathsf{pm}]}$
25 $\boxed{\sigma' \leftarrow \mathsf{EC}[H_0].\mathsf{Sign}(\mathsf{sk}, m')}$
26 $\boxed{\sigma \leftarrow \mathsf{Trf}[H_0, \mathsf{H}_1]_{\mathsf{EC}}(\mathsf{pm}, m', \sigma', \rho^{-1}, \mathsf{pk}', pk)}$
27 $\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{\mathsf{pm}\}$
28 Return $\sigma$

main in $G_4$
29 $\mathsf{RList} \leftarrow \{\epsilon\}$
30 $\mathsf{bad} \leftarrow \mathsf{false}$
31 $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{MREC}[\mathsf{H}_1].\mathsf{Gen}(\mathsf{par})$
32 $(m^*, \sigma^*, \rho^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{H}_1, \mathtt{Rand}, \mathtt{RSign}}(\mathsf{pk})$
33 $\mathsf{pk}^* \leftarrow \mathsf{pk} \cdot \rho^*$
34 $\boxed{\mathsf{pm}^* \leftarrow (\mathsf{pk}^*, m^*)}$
35 $\boxed{\text{If } H_1'[\mathsf{pm}^*] = \epsilon\text{: Abort}}$
36 If $\mathsf{pm}^* \in \mathsf{SigList}$: $\mathsf{bad} \leftarrow \mathsf{true}$
37 If $\rho^* \notin \mathsf{RList}$: $\mathsf{bad} \leftarrow \mathsf{true}$
38 $b \leftarrow \mathsf{MREC}[\mathsf{H}_1].\mathsf{Verify}(\mathsf{pk}^*, \sigma^*, m^*)$
39 Return $b \wedge \neg\mathsf{bad}$
```

Figure 10: Games G_1-G_4

B. BIP32-Compatible Threshold Wallets

In this chapter, we present the following work with minor changes:

[66] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “BIP32-Compatible Threshold Wallets”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 312. **Part of this thesis.**

BIP32-Compatible Threshold Wallets

Poulami Das [2] Andreas Erwig [1] Sebastian Faust [1] Julian Loss [2]
Siavash Riahi [1]

[1] TU Darmstadt, Germany
firstname.lastname@tu-darmstadt.de
[2] CISPA Helmholtz Center of Information Security, Germany
lossjulian@gmail.com
poulami.das@cispa.de

Abstract

Cryptographic wallets have become an essential tool to secure users' secret keys and consequently their funds in Blockchain networks. The most prominent wallet standard that is widely adopted in practice is the BIP32 specification. This standard specifies so-called hierarchical deterministic wallets, which are organized in a tree-like structure such that each node in the tree represents a wallet instance and such that a parent node can derive a new child node in a deterministic fashion. BIP32 considers two types of child nodes, namely non-hardened and hardened nodes, which differ in the security guarantees they provide. While the corruption of a hardened wallet does not affect the security of any other wallet instance in the tree, the corruption of a non-hardened node leads to a breach of the entire scheme.

In this work, we address this significant drawback of non-hardened nodes by laying out the design for the first hierarchical deterministic wallet scheme with thresholdized non-hardened nodes. We first provide a game-based notion of threshold signatures with rerandomizable keys and show an instantiation via the Gennaro and Goldfeder threshold ECDSA scheme (CCS'18). We further observe that the derivation of hardened child wallets according to the BIP32 specification does not translate easily to the threshold setting. Therefore, we devise a new and efficient derivation mechanism for hardened wallets in the threshold setting that satisfies the same properties as the original BIP32 derivation mechanism and therefore allows for efficient constructions of BIP32-compatible threshold wallets.

1 Introduction

Blockchain technologies gained huge popularity in the past few years as they provide a new decentralized mechanism to process payments without relying on a centralized authority. The main cryptographic building block in virtually all Blockchains is a digital signature scheme, which allows parties in Blockchain networks to authenticate transactions. As an example, if Alice wishes to make a payment to Bob via the Blockchain, she can sign a transaction specifying her address which is derived from her signing public key pk_A, Bob's address which is derived from his public key pk_B and the amount being spent. Alice can then sign the transaction using her secret key sk_A, so that the final transaction has the form "(pk_A pays c coins to pk_B), σ" where σ is Alice's signature on the transaction under her secret key sk_A. Essentially, Alice's secret key allows to spend all of her funds, which makes it crucial for users in a Blockchain network to protect their secret keys from attackers. In order to do so, a cryptocurrency wallet guarantees the secure storage and maintenance of a user's signing keys. While there were several different proposals for such wallet schemes [MPs19, AGKK19, KMOS21], the most widely used in practice is the BIP32 specification [Wik18], which outlines the design of *hierarchical deterministic wallets*.

Hierarchical Deterministic Wallets A hierarchical deterministic wallet scheme is organized in a tree-like structure, where one root wallet deterministically derives child wallets which in turn deterministically derive further child wallets. Each wallet in the tree is identified by an ID and consists of a signing secret and public key pair $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{pk}_{\mathsf{ID}})$ as well as a so-called chaincode $\mathsf{ch}_{\mathsf{ID}}$. BIP32 considers two types of child wallets, non-hardened and hardened child wallets, which differ only in how they are derived from the parent. More concretely, a parent wallet identified by ID derives a non-hardened wallet with identifier ID' by first computing $(\rho, \mathsf{ch}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$ and then $\mathsf{sk}_{\mathsf{ID}'} \leftarrow \mathsf{sk}_{\mathsf{ID}} + \rho$ and $\mathsf{pk}_{\mathsf{ID}'} \leftarrow \mathsf{pk}_{\mathsf{ID}} \cdot g^{\rho}$ where H is a cryptographic hash function. The values $(\mathsf{sk}_{\mathsf{ID}'}, \mathsf{pk}_{\mathsf{ID}'})$ and $\mathsf{ch}_{\mathsf{ID}'}$ then form the key pair and chaincode of the non-hardened child wallet. In contrast, the derivation of a hardened wallet differs from the above in the sense that the computation of ρ and $\mathsf{ch}_{\mathsf{ID}'}$ is carried out as $(\rho, \mathsf{ch}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$, i.e., it uses the parent's secret key $\mathsf{sk}_{\mathsf{ID}}$ as input to the hash function evaluation instead of its public key $\mathsf{pk}_{\mathsf{ID}}$. BIP32 considers these two different types of child wallets as a trade-off between usability and security. On the one hand, the derivation of a hardened wallet's public key requires knowledge of the parent secret key $\mathsf{sk}_{\mathsf{ID}}$, but provides strong security guarantees since its corruption does not affect the security of the parent node and the remaining wallets in the tree. On the other hand, the derivation of a non-hardened wallet's public key can be done by any party knowing $\mathsf{pk}_{\mathsf{ID}}$ and $\mathsf{ch}_{\mathsf{ID}}$, i.e., without knowledge of the parent secret key. This is particularly useful in the hot/cold setting (see below), where the secret key is stored in an offline device and therefore not accessible at any time. However, the corruption of a single non-hardened wallet breaks the security of the entire wallet scheme, which poses a significant security risk and severely restricts the usage of non-hardened wallets in practice.

Hot/Cold Setting Hierarchical deterministic wallets were first formally modeled and analyzed by Das et al. [DEF+21]. In their work, the authors propose to implement non-hardened wallets in the hot/cold wallet setting in order to mitigate the risk of non-hardened wallets being corrupted. In this setting, a non-hardened wallet consists of two devices, a hot wallet and a cold wallet. The hot wallet is permanently connected to the Internet and stores the public key and the chaincode whereas the cold wallet stores the secret key and the chaincode and remains offline most of the time. The assumption is then that an adversary cannot corrupt the cold wallet (and therefore does not learn the secret key) since it is mostly offline. However, this assumption might not hold where, e.g., an adversary obtains physical access to the cold wallet. Therefore, the following natural question arises:

Can we construct a secure BIP32 wallet scheme that does not rely on the idealized assumption of incorruptible cold wallets?

1.1 Our Contribution

In this work, we answer the above question affirmatively. To this end, we consider thresholdizing non-hardened nodes, s.t. each node consists of several devices where each of them stores a *share* of the signing secret key. This design choice allows to guarantee security even if a subset of the devices are corrupted. Our idea is to instantiate non-hardened wallets with a (t, n)-threshold signature scheme such that each non-hardened wallet is "split" into n different devices, each of which stores only a share of the signing secret key. As we are using a (t, n)-threshold signature scheme, at least $t + 1$ devices are required to sign a message. Simultaneously, the secret key of the non-hardened wallet remains secure as long as at most t devices are corrupted. All n devices store the public key and chaincode, s.t. only a single device can derive a non-hardened child public key without having to interact with the remaining $n - 1$ devices.

From an application perspective such a thresholdized wallet scheme can be used as a core building block for strengthening the security of so-called custodial (e.g., Coinbase [1] or BitGo [2]), self-custodial (e.g., Electrum or Trezor [Ele13, Tre14]), or shared-custodial wallets (e.g., ZenGo [3] or Sepior [4]). Custodial wallets are maintained solely by a service provider on behalf of a user, while a self-custodial wallet is maintained completely by the user itself. Naturally, while the former requires the user to trust the service provider completely, the latter is cumbersome to use as the user must possess the technical expertise to securely store its signing keys. As a trade-off between custodial and self-custodial wallets, the concept of shared-custodial wallets offers a solution where the secret key is shared among both the user and a service provider such that none can generate a valid signature without the other. For instance, a simple shared-custodial wallet can be instantiated from a $(2, 2)$-threshold wallet scheme. Of course, this can be extended to higher threshold parameters with our (t, n) construction. Due to these various useful

[1] https://custody.coinbase.com/
[2] https://www.bitgo.com/
[3] https://zengo.com/
[4] https://sepior.com/

applications, there is an increasing interest in BIP32-compatible threshold wallets [Yeh23, CHL23]. Let us now summarize our contributions in more details.

Threshold Signature Schemes with Rerandomizable Keys Das et al. [DEF$^+$21] showed that one can generically construct hierarchical deterministic wallets from signature schemes with rerandomizable keys. Such signature schemes allow to deterministically rerandomize the secret/public key pair of a signature scheme such that the rerandomized key pair constitutes again a valid signing key pair. In our threshold setting, we therefore require a threshold signature scheme with rerandomizable keys. To this end, we first provide a game-based definition of such a primitive and then show an instantiation based on the threshold ECDSA scheme of Gennaro and Goldfeder [GG18]. Importantly, for our instantiation we devise public and secret key rerandomization algorithms which allow to rerandomize the respective keys *non-interactively*, i.e., parties do not have to communicate in order to rerandomize the scheme's public key and their respective secret key shares. This is an important property for wallet schemes as we generally aim to minimize communication between wallet devices. We intentionally choose the threshold ECDSA scheme of Gennaro and Goldfeder for our instantiation for the following two reasons: (1) it is a relatively simple scheme, i.e., it does not include advanced features such as offline signing or proactive/adaptive security which significantly increase the complexity of other threshold ECDSA schemes; (2) several threshold ECDSA schemes directly build upon the protocol of Gennaro and Goldfeder [CGG$^+$20a],[DMZ$^+$21],[CCL$^+$20],[CCL$^+$21], improving either its efficiency, functionality, or security. Since the general idea of these schemes is similar to the original scheme of Gennaro and Goldfeder, we believe that our results can be extended to these schemes as well. We leave exploring such extensions as an interesting direction for future work.

(Non-)Hardened Node Derivation As a second step, we translate the (non-)hardened derivation mechanisms as specified by BIP32 to the threshold setting. To this end, we first observe that using the public and secret key rerandomization algorithms of our rerandomizable threshold ECDSA scheme we can derive non-hardened nodes according to BIP32. However, in order to securely send the rerandomized secret key shares from the parent to the child node devices, we inherently require a communication heavy protocol that re-shares the key shares from the parent to the child. We then show that the hardened node derivation cannot easily be translated to the threshold setting due to the following issue: As mentioned above, the hardened node derivation in BIP32 requires to compute a hash function evaluation on input the secret key of the parent node (and some additional inputs). In the threshold setting, however, the secret key is shared among n devices and hence, adhering to the hardened node derivation of BIP32 would require all n devices to run an interactive multi-party computation (MPC) protocol which securely computes this hash function evaluation. This is however highly inefficient and hence not a practical solution. Our main technical contribution is to provide an alternative and highly efficient hardened node derivation mechanism that still satisfies all properties of the hardened node derivation as specified by BIP32.

Our mechanism uses a (non-interactive) threshold verifiable random function (TVRF) [Dod03], which allows the n non-hardened node devices to deterministically and efficiently compute a pseudorandom value. This value can be used by the hardened node as input to the key generation algorithm of a (non-threshold) signature scheme to deterministically generate its keys. However, for this approach each non-hardened wallet instance in the tree must maintain two secret/public key pairs: one for the threshold signing scheme and one for the TVRF scheme. Similarly to the signing key pair, the TVRF keys must be deterministically derived throughout the entire tree. That is, during the derivation of a non-hardened node, the parent must re-share its signing *and* its TVRF keys, which introduces a significant communication overhead, considering that we essentially double the amount of required communication for *each* non-hardened node derivation. We would like to emphasize that the TVRF keys are only required for the derivation of hardened nodes. Yet, in practice most non-hardened wallets never derive a hardened child node and therefore would never make use of the TVRF keys, essentially wasting the communication required to derive the keys.

Due to this reason, our idea to overcome the drawback of maintaining and deriving a second key pair is to re-use the signing key pair of non-hardened nodes for the TVRF. While it is usually not recommended to re-use the same secret key over multiple cryptographic primitives, we prove that in our concrete case re-using the same secret key does not compromise security. This constitutes the main technical contribution of our work. To this end, we first formally define security properties for the joint threshold signature/TVRF scheme, and we then prove that the combined scheme satisfies our properties. The main challenge in our proof is that we must reduce the security of the joint scheme to the security of the underlying TVRF scheme. The difficulty here is that an adversary against the joint scheme is allowed to receive signatures under the schemes' secret key, while the reduction to the TVRF security does not obtain access to a signing oracle. The reduction therefore must simulate the signing protocol to the adversary in the joint scheme without having access to a signing oracle itself.

1.2 Related Work

Cryptographic Wallets There has been a plethora of works on cryptographic wallets such as [MPs19, AGKK19, CEV14, KMOS21]. We focus on hierarchical deterministic wallets, which have been extensively researched in the past. Gutoski and Stebila [GS15] introduced a hierarchical deterministic wallet scheme that, however, deviates from the BIP32 standard. Later, Das et al. [DFL19] gave the first formal analysis of deterministic wallets in the hot/cold setting and provided a construction based on multiplicatively rerandomizable ECDSA. The model of deterministic wallets by Das et al. [DFL19] has been extended to the post-quantum setting by Alkadri et al. [ADE+20]. Luzio et al. [LFA20] presented a hierarchical deterministic wallet scheme, which is however not compatible with Bitcoin. The most relevant work for our results is the paper by Das et al. [DEF+21] which analyzes the security of hierarchical deterministic wallets that comply with the BIP32 standard. As mentioned previously, Das et al. show that such wallets can be constructed generically from signature schemes with rerandomizable keys. Recently, Yin et al. [YLY+22] proposed a model for hierarchical deterministic wallets supporting stealth addresses. However, their construction is incompatible with Bitcoin as it relies on bilinear maps. Erwig and Riahi [ER22] recently proposed deterministic wallets with support for adaptor signatures and finally, in a very recent work, Chuang et al. [CHL23] investigated how the BIP32 specification can be translated to the two-party setting. That is, Chuang et al. presented two-party protocols for the key generation and hardened key derivation as specified by BIP32, and subsequently implemented these protocols and evaluated their efficiency. However, the work of Chuang et al. is restricted to the two-party setting, whereas we investigate the more general (t, n) setting.

Threshold ECDSA In recent years, there has been huge interest on threshold ECDSA (e.g., [Lin17, LN18, CGG+20a, DMZ+21, CCL+20, CCL+21, BMP22]). For a more in-depth comparison of different threshold ECDSA schemes, we refer to the survey of Aumasson et al. [AHS20]. As mentioned above, our work is based on the threshold ECDSA scheme of Gennaro and Goldfeder [GG18]. A recent work by Groth and Shoup [GS22] introduced a threshold ECDSA scheme with additive key rerandomization according to the BIP32 specification. However, the authors do not consider the derivation of hardened nodes in the threshold setting, which is the main focus of our paper. Groth and Shoup analyze their scheme in the ideal/real world setting w.r.t. an ECDSA-specifc functionality, whereas we give a general game-based definition for threshold signature schemes with rerandomizable keys and show that the construction of Gennaro and Goldfeder [GG18] can be extended to satisfy our definition. Finally, their scheme is rather complex, whereas we are aiming for a simple threshold ECDSA scheme.

2 Preliminaries

Notation. We use $s \xleftarrow{\$} H$ to denote the uniform random sampling of a value s from a set H. By $[l]$ for an integer l, we denote the set of integers $\{1, \cdots, l\}$ and for an algorithm A, we denote by $y \leftarrow A(x)$ the execution of A on input x that outputs y. We use the notation $y \in A(x)$ to denote that y is an element in the set of possible outputs of an execution of A on input x. Throughout our paper, we often avoid explicitly specifying public parameters par. Given two strings a and b, we write $a = (b, \cdot)$ if b is a prefix of a. For a set of n parties $\{P_1, \cdots, P_n\}$ and an interactive algorithm Π, we denote by $\langle \Pi(x_1), \cdots, \Pi(x_n) \rangle$ the joint execution of Π by all parties P_i for $i \in [n]$ with respective inputs x_i.

2.1 Interactive Threshold Signature Scheme

In the following, we recall the definition of interactive threshold signature schemes.

Definition 2.1 (Interactive Threshold Signature Scheme). An interactive (t, n)-threshold signature scheme TSig is executed among a set of n parties $\{P_1, \cdots, P_n\}$ and consists of a tuple of procedures $\mathsf{TSig} = (\mathsf{Gen}, \mathsf{TSign}, \mathsf{Verify})$ which are defined as follows:

- $\mathsf{Gen}(1^\kappa, t, n)$: The probabilistic key generation algorithm takes as input a security parameter κ and two integers $t, n \in \mathbb{N}$ such that $t < n$. It outputs a public key pk and a set of secret key shares $\{\mathsf{sk}_1, \cdots \mathsf{sk}_n\}$ such that each party P_i obtains pk and sk_i.
- $\mathsf{TSign}(\mathsf{sk}_i, m)$: The probabilistic signing procedure takes as input a secret key share sk_i for $i \in [n]$ and a message m. It outputs either a signature σ or $\perp$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: The deterministic verification algorithm takes as input a public key pk, a message m and a signature σ and outputs a bit 0/1.

Correctness. An interactive (t, n)-threshold signature scheme TSig is correct if for all $\kappa \in \mathbb{N}$, all $t, n \in \mathbb{N}$ with $t < n$, all $(\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n)$, and all $m \in \{0, 1\}^*$, it holds that $\Pr[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1]$, where $\sigma \leftarrow \langle \mathsf{TSign}(\mathsf{sk}_1, m), \cdots, \mathsf{TSign}(\mathsf{sk}_n, m)\rangle = 1$.

Definition 2.2 (Unforgeability of interactive threshold signature schemes). An interactive (t, n)-threshold signature scheme TSig is unforgeable if no PPT adversary $\mathcal{A}$ wins game **th-ufcma** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{th-ufcma}_{\mathsf{TSig}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{th-ufcma}_{\mathsf{TSig}}} := \Pr[\textbf{th-ufcma}^{\mathcal{A}}_{\mathsf{TSig}} = 1]$.

Game $\textbf{th-ufcma}_{\mathsf{TSig}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \leq t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.
- The game initializes a list $\mathsf{SigList} \leftarrow \{\epsilon\}$ and executes $(\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}, \mathsf{pk}) \leftarrow \mathsf{TSig.Gen}(1^\kappa, t, n)$. Then $\mathcal{A}$ is run on input pk and $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$.
- The adversary obtains access to the following `Sign` oracle: On input message m, the oracle and the adversary jointly execute the procedure $\mathsf{TSig.TSign}$, where the oracle runs all honest parties P_i on input (sk_i, m). The message m is then stored in $\mathsf{SigList}$.
- Eventually, the adversary outputs a forgery σ^* and a message m^*. The adversary wins the game, if the following conditions hold: (1) $\mathsf{TSig.Verify}(\mathsf{pk}^*, m^*, \sigma^*) = 1$ and (2) $m^* \notin \mathsf{SigList}$.

2.2 Signature Scheme with Honestly Rerandomizable Keys

The notion of signature schemes with rerandomizable keys has first been introduced by Fleischhacker et al. [FKM+16].

Definition 2.3 (Signature Scheme with Perfectly Rerandomizable Keys). Let the public parameters par define a randomness space $\mathcal{R} := \mathcal{R}(\mathsf{par})$. A signature scheme with perfectly rerandomizable keys is then a tuple of algorithms $\mathsf{RSig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RandSK}, \mathsf{RandPK})$ where $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ are the standard algorithms of a signature scheme. The algorithms RandSK and RandPK are defined as follows:

- $\mathsf{RandSK}(\mathsf{sk}, \rho)$: The deterministic *secret key rerandomization algorithm* RandSK takes as input a secret key sk and randomness $\rho \in \mathcal{R}$ and outputs a rerandomized secret key sk'.
- $\mathsf{RandPK}(\mathsf{pk}, \rho)$: The deterministic *public key rerandomization algorithm* RandPK takes as input a public key pk and randomness $\rho \in \mathcal{R}$ and outputs a rerandomized public key pk'.

We recall the correctness definition and the security notion of *one-per message existential unforgeability under honestly rerandomizable keys* (**uf-cma-hrk1**) for signature schemes with rerandomizable keys [DEF+21] in Appendix A.1. Further, we recall the construction of an additively rerandomizable and **uf-cma-hrk1**-secure ECDSA signature scheme as presented by Das et al. [DEF+21] in Appendix A.3.

2.3 Non-Interactive Threshold Verifiable Random Function

We recall the definition of a non-interactive threshold verifiable random function from Galindo et al. [GLOW21].[5]

Definition 2.4 A non-interactive (t,n)-threshold verifiable random function (TVRF) is defined w.r.t. to a randomness space Rand and is executed among n parties $\{P_1,\cdots,P_n\}$. It consists of a tuple of algorithms $\mathsf{TVRF} = (\mathsf{Gen},\mathsf{PEval},\mathsf{Combine},\mathsf{Verify})$ which are defined as follows:

- $\mathsf{Gen}(1^\kappa,t,n)$: The probabilistic key generation algorithm Gen takes as input a security parameter κ and two integers $t,n \in \mathbb{N}$ such that $t < n$. It outputs a public key pk and a set of secret key shares $\{\mathsf{sk}_1,\cdots\mathsf{sk}_n\}$ such that each party P_i obtains pk and sk_i.
- $\mathsf{PEval}(m,\mathsf{sk}_i,\mathsf{pk})$: The partial evaluation algorithm PEval takes as input a message m, a secret key share sk_i, and a public key pk, and it outputs an evaluation share ϕ_i and a proof π_i.
- $\mathsf{Combine}(\mathsf{pk},m,\mathcal{S},\{\phi_i,\pi_i\}_{i\in\mathcal{S}})$: The combination algorithm $\mathsf{Combine}$ takes as input a public key pk, a message m, a set of indices $\mathcal{S}$ with $|\mathcal{S}| > t$, and a set of partial evaluation shares $\{\phi_i,\pi_i\}_{i\in\mathcal{S}}$. It outputs either a function evaluation $\phi \in \mathsf{Rand}$ and a proof π, or $\perp$.
- $\mathsf{Verify}(\mathsf{pk},m,\phi,\pi)$: The verification algorithm Verify takes as input a public key pk, a message m, a function evaluation $\phi \in \mathsf{Rand}$, and a proof π, and outputs either 0 or 1.

A TVRF must satisfy the properties *uniqueness*, *pseudorandomness*, and *robustness*.

Definition 2.5 (Uniqueness of TVRF). A non-interactive (t,n)-threshold verifiable random function scheme TVRF is **th-unique**-secure if no PPT adversary $\mathcal{A}$ wins game **th-unique** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{th-unique}_{\mathsf{TVRF}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{th-unique}_{\mathsf{TVRF}}} := \Pr[\textbf{th-unique}^{\mathcal{A}}_{\mathsf{TVRF}} = 1]$.

Game $\textbf{th-unique}_{\mathsf{TVRF}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \le t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.
- The game executes $(\mathsf{pk},\{\mathsf{sk}_1,\cdots,\mathsf{sk}_n\}) \leftarrow \mathsf{TVRF.Gen}(1^\kappa,t,n)$. $\mathcal{A}$ receives as input pk and $\{\mathsf{sk}_i\}_{i\in\mathsf{C}}$.
- The adversary obtains access to the following oracles:
  - `Eval`: On input message m and index $i \in [n]\setminus\mathsf{C}$, the oracle executes $(\phi_i,\pi_i) \leftarrow \mathsf{TVRF.PEval}(m,\mathsf{sk}_i,\mathsf{pk})$ and returns (ϕ_i,π_i).
  - `KeyLeak`: On input an index $i \in [n]$, the oracle outputs sk_i.
- Eventually, the adversary outputs a message m^* and two function evaluations $\{(\phi^{i^*},\pi^{i^*})\}_{i\in\{0,1\}}$. The game outputs 1 if $\phi^{0^*} \neq \phi^{1^*}$ and $\mathsf{TVRF.Verify}(\mathsf{pk},m^*,\phi^{0^*},\pi^{0^*}) = \mathsf{TVRF.Verify}(\mathsf{pk},m^*,\phi^{1^*},\pi^{1^*}) = 1$. Otherwise it outputs 0.

Definition 2.6 (Pseudorandomness of TVRF). A non-interactive (t,n)-threshold verifiable random function scheme TVRF is **th-prand**-secure if no PPT adversary $\mathcal{A}$ wins game **th-prand** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{th-prand}_{\mathsf{TVRF}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{th-prand}_{\mathsf{TVRF}}} := \Pr[\textbf{th-prand}^{\mathcal{A}}_{\mathsf{TVRF}} = 1] - \frac{1}{2}$.

Game $\textbf{th-prand}_{\mathsf{TVRF}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \le t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.

[5]We note that Galindo et al. refer to the primitive in their work as *non-interactive fully distributed verifiable random function.*

- The game initializes $\mathsf{EvalList} \leftarrow \{\epsilon\}$ and executes $(\mathsf{pk}, \{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}) \leftarrow \mathsf{TVRF.Gen}(1^\kappa, t, n)$. $\mathcal{A}$ receives as input pk and $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$.
- The adversary obtains access to the following oracle:
  - `Eval`: On input message m and an index $i \in [n] \setminus \mathsf{C}$, the oracle executes $(\phi_i, \pi_i) \leftarrow \mathsf{TVRF.PEval}(m, \mathsf{sk}_i, \mathsf{pk})$ and if $(i, m) \notin \mathsf{EvalList}$, stores the tuple (i, m) in $\mathsf{EvalList}$. The oracle returns (ϕ_i, π_i).
- Eventually, the adversary outputs a message m^*, a set of indices $\mathcal{S}$ with $\mathcal{S} > t$, evaluation shares $\{\phi_i, \pi_i\}_{i \in \mathcal{S} \cap \mathsf{C}}$. The game checks if there are less than $t - |\mathcal{S} \cap \mathsf{C}|$ tuples of the form $(\cdot, m^*)$ in $\mathsf{EvalList}$ and if so, the game computes for $j \in \mathcal{S} \setminus \mathsf{C}$ the tuple $(\phi_j, \pi_j) \leftarrow \mathsf{TVRF.PEval}(m^*, \mathsf{sk}_j, \mathsf{pk})$ and $(\phi, \pi) \leftarrow \mathsf{TVRF.Combine}(\mathsf{pk}, \mathcal{S}, \{(\phi_i, \pi_i)\}_{i \in \mathcal{S}})$. If $\phi = \perp$, the game returns ϕ. Otherwise the game chooses a bit $b \xleftarrow{\$} \{0, 1\}$ and does the following:
  - If $b = 0$: Return ϕ.
  - If $b = 1$: Sample $\phi' \xleftarrow{\$} \mathsf{Rand}$ and output ϕ'.

  The adversary then outputs a bit b' and wins if $b = b'$.

Definition 2.7 (Robustness of TVRF). A non-interactive (t, n)-threshold verifiable random function scheme TVRF is **th-robust**-secure if no PPT adversary $\mathcal{A}$ wins game **th-robust** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{th-robust}_{\mathsf{TVRF}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{th-robust}_{\mathsf{TVRF}}} := \Pr[\textbf{th-robust}^{\mathcal{A}}_{\mathsf{TVRF}} = 1]$.

Game $\textbf{th-robust}_{\mathsf{TVRF}}$:

- The game differs from game $\textbf{th-prand}_{\mathsf{TVRF}}$ only by the winning condition, which we will describe below.
- The adversary outputs a message m^*, a set $\mathcal{S}$ with $|\mathcal{S}| > t$ and a set of evaluation shares $\{\phi_i, \pi_i\}_{i \in \mathcal{S} \cap \mathsf{C}}$. The game computes $(\phi_i, \pi_i) \leftarrow \mathsf{TVRF.PEval}(m^*, \mathsf{sk}_i, \mathsf{pk})$ for all $i \in \mathcal{S} \setminus \mathsf{C}$. The game finally sets $(\phi^*, \pi^*) \leftarrow \mathsf{TVRF.Combine}(\mathsf{pk}, \mathcal{S}, \{\phi_i, \pi_i\}_{i \in \mathcal{S}})$. If $\phi^* \neq \perp$ and $\mathsf{TVRF.Verify}(\mathsf{pk}, m^*, \phi^*, \pi^*) = 0$, the game outputs 1 and 0 otherwise.

3 Rerandomizable Interactive Threshold Signing

3.1 Model

In the following, we introduce the notion of interactive threshold signature schemes with rerandomizable keys. More concretely, we extend the standard notion of a threshold signature scheme by two algorithms RandSK and RandPK, which allow to individually derive rerandomized secret key shares and a rerandomized public key respectively, such that the derived secret key shares form a valid (t, n)-sharing of the secret key that corresponds to the derived public key.

Definition 3.1 (Interactive Threshold Signature Scheme With Rerandomizable Keys). An interactive (t, n)-threshold signature scheme with rerandomizable keys is a tuple of procedures $\mathsf{RTSig} = (\mathsf{Gen}, \mathsf{RandSK}, \mathsf{RandPK}, \mathsf{TSign}, \mathsf{Verify})$ where $(\mathsf{Gen}, \mathsf{TSign}, \mathsf{Verify})$ are defined as for interactive (t, n)-threshold signatures. We assume that the public parameters par define a randomness space $\mathcal{R} := \mathcal{R}(\mathsf{par})$. The algorithms RandSK and RandPK are defined as:

- $\mathsf{RandSK}(i, \mathsf{sk}_i, \rho)$: The deterministic secret key share rerandomization algorithm takes as input an index $i \in [n]$, a secret key share sk_i and a randomness $\rho \in \mathcal{R}$ and it outputs a rerandomized secret key share sk'_i.
- $\mathsf{RandPK}(\mathsf{pk}, \rho)$: The deterministic public key rerandomization algorithm takes as input a public key pk and a randomness $\rho \in \mathcal{R}$ and it outputs a rerandomized public key pk'.

We require the following properties of a threshold signature scheme with rerandomizable keys:

- *Rerandomizability of public keys*: For all $\kappa \in \mathbb{N}$, all $t, n \in \mathbb{N}$ with $t < n$, all $(\cdot, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n)$ and all $\rho \xleftarrow{\$} \mathcal{R}$, the distributions of pk' and pk'' are computationally indistinguishable, where $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$ and $(\cdot, \mathsf{pk}'') \leftarrow \mathsf{Gen}(1^\kappa, t, n)$.

- *Correctness under rerandomized keys*: For all $\kappa \in \mathbb{N}$, all $t, n \in \mathbb{N}$ with $t < n$, all $(\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n)$, all $\rho \xleftarrow{\$} \mathcal{R}$ and all $m \in \{0,1\}^*$, the rerandomized keys $\{\mathsf{sk}'_i\}_{i \in [n]} \leftarrow \{\mathsf{RandSK}(i, \mathsf{sk}_i, \rho)\}_{i \in [n]}$ and $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$ satisfy:

$$\Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma) | \sigma \leftarrow \langle \mathsf{TSign}(\mathsf{sk}'_1, m), \cdots, \mathsf{TSign}(\mathsf{sk}'_n, m) \rangle] = 1$$

We note that the property of rerandomizability of public keys is a slightly weaker notion than the perfect rerandomizability of keys of rerandomizable signature schemes (cf. Appendix A) which requires rerandomized public and secret keys to be identically distributed to a freshly generated key pair. However, as previously pointed out by Alkadri et al. [ADE+20], this weaker rerandomizability property is sufficient for the wallet setting. At a high level, that is because this notion is required to ensure the wallet unlinkability property, which guarantees unlinkability of wallet *public keys*, i.e., it guarantees that a derived public key is computationally indistinguishable from freshly generated public keys.

We define the security notion of *one-per message existential unforgeability under honestly rerandomizable keys* for interactive threshold signature schemes with rerandomizable keys. That is, we define a security game **th-ufcma-hrk1** which differs from the unforgeability game **th-ufcma** (cf. Def. 2.2) of interactive threshold signatures in the following ways: (1) the adversary receives access to a `Rand` oracle, which outputs uniformly random elements from $\mathcal{R}$; (2) the signing oracle `RSign` cannot only generate signatures under the initial set of keys $(\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}, \mathsf{pk})$, but also under key sets that have been rerandomized with an element output by the `Rand` oracle; (3) the signing oracle returns at most one signature for each key set/message pair; and (4) the adversary can win the game with a valid forgery under *any* key set rerandomized with an output of the `Rand` oracle. We note that the notion of one-per message unforgeability is weaker than standard unforgeability, however, as remarked by Das et al. [DEF+21] this weaker notion is sufficient for the wallet setting.

Definition 3.2 (One-per message unforgeability of interactive threshold signature schemes with honestly rerandomizable keys). An interactive (t, n)-threshold signature scheme with rerandomizable keys $\mathsf{RTSig} = (\mathsf{Gen}, \mathsf{RandSK}, \mathsf{RandPK}, \mathsf{TSign}, \mathsf{Verify})$ is **th-ufcma-hrk1**-secure if no PPT adversary $\mathcal{A}$ wins game **th-ufcma-hrk1** as described below with more than negligible probability in the security parameter κ.

Game **th-ufcma-hrk1**$_{\mathsf{RTSig}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \leq t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.

- The game initializes two lists $\mathsf{RList} \leftarrow \{\epsilon\}$ and $\mathsf{SigList} \leftarrow \{\epsilon\}$ and executes $(\{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}, \mathsf{pk}) \leftarrow \mathsf{RTSig.Gen}(1^\kappa, t, n)$. Then, $\mathcal{A}$ is run on input pk and $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$.

- The adversary obtains access to the following two oracles:

  – `Rand`: This oracle, upon a query, samples $\rho \xleftarrow{\$} \mathcal{R}$, stores ρ in RList and outputs ρ to $\mathcal{A}$.

  – `RSign`: On input message m and a randomness ρ, the oracle checks whether $\rho \notin \mathsf{RList}$ and if so outputs $\perp$. Otherwise, it derives a public key and secret key shares for honest parties with the randomness ρ, i.e., it computes $\mathsf{pk}' \leftarrow \mathsf{RTSig.RandPK}(\mathsf{pk}, \rho)$ and $\mathsf{sk}'_i \leftarrow \mathsf{RTSig.RandSK}(i, \mathsf{sk}_i, \rho)$ for all $i \in \{1, \cdots, n\} \setminus \mathsf{C}$. If $(\mathsf{pk}', m) \in \mathsf{SigList}$ then the oracle returns $\perp$. Otherwise, the oracle and the adversary jointly execute the procedure $\mathsf{RTSig.TSign}$, where the oracle runs all honest parties P_i on input (sk'_i, m). The oracle then stores the tuple (pk', m) in $\mathsf{SigList}$.

- Eventually, the adversary outputs a forgery σ^*, a message m^* and a public key $\mathsf{pk}^* \leftarrow \mathsf{RTSig.RandPK}(\mathsf{pk}, \rho^*)$. The adversary wins the game, if the following conditions hold: (1) $\rho^* \in \mathsf{RList}$, (2) $(\mathsf{pk}^*, m^*) \notin \mathsf{SigList}$, and (3) $\mathsf{RTSig.Verify}(\mathsf{pk}^*, m^*, \sigma^*) = 1$.

3.2 Construction

We show how to extend the interactive threshold ECDSA scheme as proposed by Gennaro and Goldfeder [GG18] (which we denote by $\mathsf{GG}[\mathsf{H}_0]$) to an interactive threshold ECDSA scheme with rerandomizable keys (which we denote by $\mathsf{rGG}[\mathsf{H}_0]$). We recall the scheme of Gennaro and Goldfeder (with a slight adjustment) in detail in Appendix B. In Figure 1, we describe our $\mathsf{rGG}[\mathsf{H}_0]$ scheme w.r.t. the $\mathsf{GG}[\mathsf{H}_0]$ scheme. Recall that the ECDSA signature scheme is defined w.r.t. a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q and that an ECDSA key pair $(\mathsf{pk}, \mathsf{sk})$ is simply computed as $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$. In the $\mathsf{GG}[\mathsf{H}_0]$ scheme, the secret key is shared such that each party P_i holds a secret key share sk_i and a public key share g^{sk_i}. In our $\mathsf{rGG}[\mathsf{H}_0]$ scheme, we extend the $\mathsf{GG}[\mathsf{H}_0]$ scheme by providing algorithms RandSK and RandPK which deterministically rerandomize the secret key shares and the public key respectively w.r.t. a randomness ρ. At a high level, in order to rerandomize the secret key share sk_i of party P_i with randomness ρ, the RandSK algorithm deterministically generates a degree-t polynomial F with coefficients in $\mathbb{Z}_q$ and evaluates the polynomial at point i. This essentially yields a randomness share ρ_i, which is then added to the existing secret key share to compute the rerandomized secret key share $\mathsf{sk}'_i \leftarrow \mathsf{sk}_i + \rho_i \mod q$. That is, sk'_i is essentially a share of the secret key $\mathsf{sk} + \rho \mod q$. The RandPK algorithm works correspondingly for the public key and public key shares.

The security of our $\mathsf{rGG}[\mathsf{H}_0]$ scheme can be proven via a reduction to the (one-per message) unforgeability of the ECDSA scheme with rerandomizable keys by Das et al. [DEF+21], which we recall in Appendix A.3. Note that the scheme of Das et al. is public key prefixed, i.e., whenever a message m is signed using secret key sk, the message is first prefixed with the corresponding public key pk, s.t. the signature is generated for (pk, m). Since we reduce the security of our $\mathsf{rGG}[\mathsf{H}_0]$ scheme to the (one-per message) unforgeability of the scheme of Das et al., we require public key prefixing in our scheme as well.

```
Algorithm $\mathsf{Gen}(1^\kappa, t, n)$
00 Return $\mathsf{GG.Gen}(1^\kappa, t, n)$

Algorithm $\mathsf{RandSK}(i, \mathsf{sk}_i, \rho)$
00 For $k \in [t] : a_k \leftarrow \mathsf{H}_0(\rho, k)$
01 Let $F(x) := a_t x^t + \cdots + a_1 x + \rho$
02 $\rho_i \leftarrow F(i) \mod q$
03 $\mathsf{sk}'_i \leftarrow \mathsf{sk}_i + \rho_i \mod q$
04 Return $\mathsf{sk}'_i$

Protocol $\mathsf{TSign}(\mathsf{sk}_i, m)$
00 $m' \leftarrow (\mathsf{pk}, m)$
01 Return $\mathsf{GG.TSign}(\mathsf{sk}_i, m')$

Algorithm $\mathsf{RandPK}(\mathsf{pk}, \rho)$
00 Parse $\mathsf{pk} := (X, (X_1, \cdots, X_n))$
01 For $k \in [t] : a_k \leftarrow \mathsf{H}_0(\rho, k)$
02 Let $F(x) := a_t x^t + \cdots + a_1 x + \rho$
03 $\rho_i \leftarrow F(i) \mod q$
04 For $i \in [n] : X'_i \leftarrow X_i \cdot g^{\rho_i}$
05 Return $\mathsf{pk}' := (X \cdot g^\rho, (X'_1, \cdots, X'_n))$

Algorithm $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
00 $m' \leftarrow (\mathsf{pk}, m)$
01 Return $\mathsf{GG.Verify}(\mathsf{pk}, m', \sigma)$
```

Figure 1: Public key prefixed interactive threshold ECDSA scheme $\mathsf{rGG}[\mathsf{H}_0]$ with honestly rerandomizable keys based on the $\mathsf{GG}[\mathsf{H}_0]$ scheme for a hash function $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_q$. For brevity, we denote scheme $\mathsf{GG}[\mathsf{H}_0]$ by GG.

It is easy to see that the $\mathsf{rGG}[\mathsf{H}_0]$ scheme satisfies the *correctness under rerandomized keys* property.

Lemma 3.3 *Let $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle and let the discrete logarithm problem be hard in $\mathbb{G}$. Then the interactive (t,n)-threshold ECDSA scheme with rerandomizable keys $\mathsf{rGG}[\mathsf{H}_0]$ satisfies the rerandomizability of public keys property.*

proof sketch. We can prove the above lemma via reduction to the discrete logarithm problem. At a high level, assume there exists a PPT adversary $\mathcal{A}$ that can distinguish the distributions

$$\{\mathsf{pk}, \mathsf{pk}' \mid (\cdot, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n), \rho \xleftarrow{\$} \mathcal{R}, \mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)\}$$
$$\text{and}$$
$$\{\mathsf{pk}, \mathsf{pk}'' \mid (\cdot, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n), (\cdot, \mathsf{pk}'') \leftarrow \mathsf{Gen}(1^\kappa, t, n)\}$$

with more than negligible probability, then we can construct a PPT adversary $\mathcal{B}$ that breaks the discrete logarithm problem with a related probability. In the following, we sketch how this reduction proceeds:

adversary $\mathcal{B}$ receives a discrete logarithm challenge g^ρ as input. $\mathcal{B}$ then samples $(\cdot, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa, t, n)$ and computes pk' by executing algorithm $\mathsf{RandPK}(\mathsf{pk}, \rho)$, however with the following differences: (1) it samples the coefficients a_k for $k \in [t]$ of polynomial F uniformly at random from $\mathbb{Z}_q$ instead of computing them as $\mathsf{H}_0(\rho, k)$; and (2) it evaluates polynomial F only in the exponent, i.e., it computes $g^{\rho_i} \leftarrow g^{F(i)}$ for $i \in [n]$.[6] Note that $\mathcal{A}$ can only detect these changes if it queries the random oracle H_0 on input (ρ, k). However, if $\mathcal{A}$ makes a query of this form, adversary $\mathcal{B}$ learns ρ, which is the discrete logarithm of its discrete logarithm challenge g^ρ. Otherwise, public key pk' is identically distributed to a freshly generated public key pk''. ■

Theorem 3.4 *Let* PKE *be a semantically secure linearly homomorphic encryption scheme,* ZK *be a non-interactive zero-knowledge proof system and* CT *a non-malleable and equivocable commitment scheme. Further, let the DDH assumption hold in $\mathbb{G}$ and let* $\mathsf{rECDSA}[\mathsf{H}_0]$ *be the* **uf-cma-hrk1**-*secure ECDSA scheme with rerandomizable keys as described in Appendix A.3. Then the interactive (t, n)-threshold ECDSA scheme with rerandomizable keys* $\mathsf{rGG}[\mathsf{H}_0]$ *as described above is* **th-ufcma-hrk1**-*secure.*

Sketch. Gennaro and Goldfeder prove the $\mathsf{GG}[\mathsf{H}_0]$ scheme unforgeable via reduction to the unforgeability of the single party ECDSA signature scheme. That is, they provide a reduction that simulates game $\textbf{th-ufcma}_{\mathsf{GG}[\mathsf{H}_0]}$ (cf. Definition 2.2) while having access to a signing oracle that outputs ECDSA signatures for adaptively chosen messages. Gennaro and Goldfeder prove that this simulation is computationally indistinguishable from the real game to a PPT adversary. We recall the simulation in Appendix B[7]. We can prove the above theorem in the same way, with the difference that we reduce the $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}$ security to the $\textbf{uf-cma-hrk1}_{\mathsf{rECDSA}[\mathsf{H}_0]}$ security. That is, we have to provide a reduction that simulates game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}$ to a PPT adversary while having access to the `RSign` and `Rand` oracles of game $\textbf{uf-cma-hrk1}_{\mathsf{rECDSA}[\mathsf{H}_0]}$. In fact, we can use the same simulation as the one from Gennaro and Goldfeder with the following differences: (1) Upon the adversary querying the `Rand` oracle in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}$, the reduction relays the query to its own `Rand` oracle in game $\textbf{uf-cma-hrk1}_{\mathsf{rECDSA}[\mathsf{H}_0]}$; (2) Upon the adversary querying oracle `RSign` in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}$ on input a message m and randomness ρ, the reduction first rerandomizes the secret key shares sk_i of corrupted parties $P_i \in \mathsf{C}$ by computing $\mathsf{sk}'_i \leftarrow \mathsf{RandSK}(i, \mathsf{sk}_i, \rho)$ as well as the public key $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$. The reduction then queries its own signing oracle on input m and ρ and uses the resulting signature and the rerandomized keys for the simulation of the `RSign` oracle of game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}$. These changes do not have any impact on the indistinguishability arguments and reduction from Gennaro and Goldfeder. Note that, since we essentially repeat the proof of Gennaro and Goldfeder, we must also repeat the assumptions their proof relies on in our theorem statement. ■

4 BIP32-Compatible Threshold Wallets

In order to construct threshold BIP32 wallets, we require two ingredients, namely (1) a threshold signature scheme with rerandomizable keys, and (2) mechanisms for the derivation of non-hardened and hardened wallets in the threshold setting. With requirement (1) in place, we will discuss in this section how the respective wallet derivations of a BIP32 wallet can be implemented in the threshold setting. In particular, we consider the following setting for our threshold BIP32 wallet: All non-hardened wallets are thresholdized, i.e., each non-hardened wallet consists of n devices which execute a (t, n)-threshold signature scheme with rerandomizable keys. We assume throughout the paper that $t \leq \frac{n-1}{2}$ for all non-hardened nodes. Hardened wallets, on the other hand, are single devices (i.e. not thresholdized), since the corruption of a hardened wallet does not affect the security of the remaining wallets in the tree. Similar to the modeling of BIP32 wallets by Das et al. [DEF+21], we do not allow hardened wallets to derive child wallets, i.e., hardened wallets always represent leaves in the wallet tree. Therefore, we assume that in both cases, i.e., the non-hardened and hardened wallet derivation, the parent wallet is non-hardened and thresholdized. Recall that BIP32 specifies the (non-threshold) derivation mechanisms as follows: A non-hardened node with identifier ID' is derived from a parent node with identifier ID, key pair $(\mathsf{sk}_{\mathsf{ID}}, \mathsf{pk}_{\mathsf{ID}})$ and chaincode $\mathsf{ch}_{\mathsf{ID}}$ by computing $(\rho, \mathsf{ch}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$, $\mathsf{sk}_{\mathsf{ID}'} \leftarrow \mathsf{sk}_{\mathsf{ID}} + \rho$ and $\mathsf{pk}_{\mathsf{ID}'} \leftarrow \mathsf{pk}_{\mathsf{ID}} \cdot g^\rho$. The derivation of a hardened node works in the same way only that the tuple $(\rho, \mathsf{ch}_{\mathsf{ID}'})$

[6]This step is necessary because $\mathcal{B}$ only knows g^ρ but not ρ. Therefore, $\mathcal{B}$ can only compute F in the exponent.

[7]To be exact, since our $\mathsf{GG}[\mathsf{H}_0]$ scheme differs slightly from the original scheme of Gennaro and Goldfeder, we recall a slightly adjusted simulation. See Appendix B for details.

is computed as $\mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$. We now analyze these derivation mechanisms for the threshold setting w.r.t. to our threshold signature scheme with rerandomizable keys $\mathsf{rGG}[\mathsf{H}_0]$ in more detail.

4.1 Non-Hardened Node Derivation

The derivation of non-hardened nodes in the threshold setting is fairly straightforward and follows the ideas of the BIP32 standard. Essentially, a non-hardened parent node identified by ID and consisting of n devices s.t. each device stores a secret key share $\mathsf{sk}_{i,\mathsf{ID}}$ and the chaincode $\mathsf{ch}_{\mathsf{ID}}$ can derive a thresholdized non-hardened child wallet as follows: First, each device of the parent node computes locally $(\rho, \mathsf{ch}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{pk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$ and $\mathsf{sk}_{i,\mathsf{ID}'} \leftarrow \mathsf{rGG}[\mathsf{H}_0].\mathsf{RandSK}(i, \mathsf{sk}_{i,\mathsf{ID}}, \rho)$. Then the devices of the parent node must forward the rerandomized secret key shares $\mathsf{sk}_{i,\mathsf{ID}'}$ and the chaincode $\mathsf{ch}_{\mathsf{ID}'}$ to the n devices of the child node. The forwarding of the chaincode $\mathsf{ch}_{\mathsf{ID}'}$ is straightforward, since we assume an honest majority among the parent devices and since each parent device knows $\mathsf{ch}_{\mathsf{ID}'}$. That is, all parent devices can simply send $\mathsf{ch}_{\mathsf{ID}'}$ to all child devices. Each child device then receives at least $t+1$ times the value $\mathsf{ch}_{\mathsf{ID}'}$ which it uses as the node's chaincode. The forwarding of the secret key shares $\mathsf{sk}_{i,\mathsf{ID}'}$ is more involved and requires a protocol involving $2n$ devices (n child and n parent wallet devices) of which a total of $2t$ devices can be corrupted. Note that a simple forwarding of secret key share $\mathsf{sk}_{i,\mathsf{ID}'}$ to the i-th device of the child wallet is insecure as it allows an adversary to learn a total of $2t$ secret key shares. Instead, the $2n$ devices must engage in the execution of a dynamic proactive secret sharing (DPSS) scheme (e.g., [BDLO15, MZW+19, SLL10]), which allows to *securely* handover the rerandomized key shares to the devices of the child node even in the presence of $2t$ corrupted devices. Note that DPSS schemes typically incur a significant communication overhead since all $2n$ parties must interact with each other.

4.2 Hardened Node Derivation

The main challenge when considering BIP32 wallets in the threshold setting is designing a derivation mechanism for hardened nodes. Recall that the derivation of a hardened node according to BIP32 requires the computation of $(\rho, \mathsf{ch}_{\mathsf{ID}'}) \leftarrow \mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$, i.e., the evaluation of a hash function on input the parent secret key. In the threshold setting, however, the secret key $\mathsf{sk}_{\mathsf{ID}}$ is shared among n devices such that no single device knows the full key. It is therefore not at all clear how $\mathsf{H}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ch}_{\mathsf{ID}}, \mathsf{ID}')$ can be computed efficiently without naively reconstructing $\mathsf{sk}_{\mathsf{ID}}$ (which would trivially break the security of the wallet). Furthermore, in the hardened derivation, each parent device can only learn a randomness *share* ρ_i instead of the entire randomness ρ. To see why that is, consider the setting where an adversary corrupts the hardened node, thereby learning its secret key $\mathsf{sk}_{\mathsf{ID}} + \rho$, as well as a parent node device, thereby learning ρ. The adversary could then trivially learn the parent node's secret key.

One obvious (and to the best of our knowledge the only) way to resolve the above issues is using generic multi-party computation (MPC) techniques [GMW87, Gol04, CCD88], which allow to securely compute any function in a distributed setting without revealing the function inputs. However, generic MPC is inherently inefficient, in particular since the BIP32 standard uses the well-known hash function SHA-512, which is known to be only inefficiently computable via MPC [BST21].

An Improved Derivation Mechanism Due to the above limitation, we consider a more efficient hardened node derivation mechanism, which achieves the same properties as the one originally specified in BIP32. We circumvent the inefficient distributed SHA-512 execution by letting the devices of the non-hardened parent wallet jointly and deterministically generate a random seed in such a way that only the hardened node but no parent device learns the seed. The hardened node can then use this seed as input to the key generation algorithm of a (non-threshold) signature scheme (ECDSA in our case) to deterministically generate its key pair. Said differently, instead of having the parent wallet devices rerandomize their secret key shares and forward them to the hardened wallet, we simply let the parent devices generate a random value from which the hardened node can deterministically derive its own keys. For the computation of the random seed, we employ the threshold verifiable random function (TVRF) from [GLOW21]. A (t, n)-TVRF is a cryptographic primitive that is executed by n parties, where each party P_i knows a secret key share sk_i, which it can use to deterministically compute an evaluation share ϕ_i and proof π_i on a message m. Given at least $t+1$ evaluation shares for m, any party can deterministically compute a pseudorandom value ϕ and a proof π and given the public key pk, ϕ and π, any party can verify that ϕ was computed correctly. We recall the formal definition of a TVRF in Section 2.3.

We use the TVRF for the hardened wallet derivation in the following way: Each device of the non-hardened parent node maintains a secret key share for the TVRF and, upon the derivation of a hardened node with identifier ID, it uses this share to compute an evaluation share ϕ_i and the corresponding proof π_i on ID. It then sends (ϕ_i, π_i) to the hardened node, which combines $t+1$ shares to a pseudorandom seed ϕ. The hardened node then verifies the correctness of ϕ using the public key of the TVRF. Note that any set of $t+1$ correct evaluation shares will yield the same seed, but including only a single invalid evaluation share will lead to a different (incorrect) seed. Therefore, the verifiability of the seed is crucial to our solution. We use the TVRF from [GLOW21] which is not only deterministic and one-way but also non-interactively computable, therefore exhibiting the same properties as the original BIP32 derivation mechanism. We present our improved hardened node derivation mechanism pictorially in Figure 2.

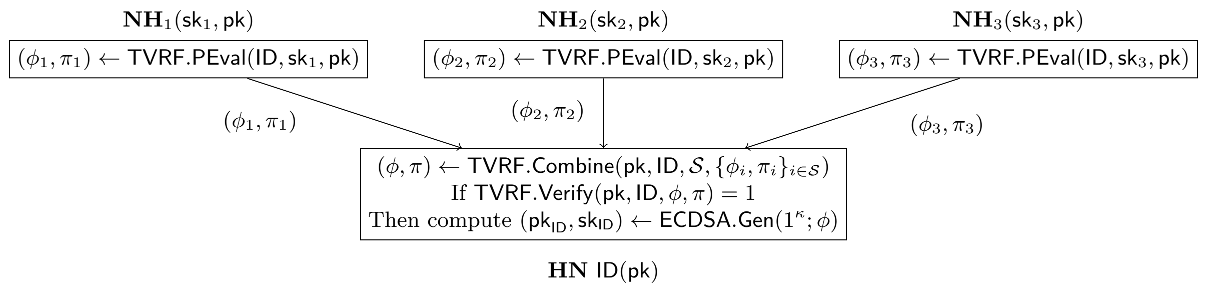

$\mathbf{NH}_1(\mathsf{sk}_1, \mathsf{pk})$
$(\phi_1, \pi_1) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_1, \mathsf{pk})$
$\mathbf{NH}_2(\mathsf{sk}_2, \mathsf{pk})$
$(\phi_2, \pi_2) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_2, \mathsf{pk})$
$\mathbf{NH}_3(\mathsf{sk}_3, \mathsf{pk})$
$(\phi_3, \pi_3) \leftarrow \mathsf{TVRF.PEval}(\mathsf{ID}, \mathsf{sk}_3, \mathsf{pk})$
(ϕ_1, π_1)
(ϕ_2, π_2)
(ϕ_3, π_3)
$(\phi, \pi) \leftarrow \mathsf{TVRF.Combine}(\mathsf{pk}, \mathsf{ID}, \mathcal{S}, \{\phi_i, \pi_i\}_{i \in \mathcal{S}})$
If $\mathsf{TVRF.Verify}(\mathsf{pk}, \mathsf{ID}, \phi, \pi) = 1$
Then compute $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}) \leftarrow \mathsf{ECDSA.Gen}(1^\kappa; \phi)$
$\mathbf{HN}$ $\mathsf{ID}(\mathsf{pk})$

Figure 2: Pictorial representation of our improved hardened node derivation mechanism in the threshold setting. Each of the three devices $\mathbf{NH}_1$, $\mathbf{NH}_2$, $\mathbf{NH}_3$ of the non-hardened parent node stores a TVRF public key pk and secret key share sk_i for $i \in [3]$. In order to derive a hardened node $\mathbf{HN}$ with identity ID, each non-hardened device locally evaluates the TVRF on input ID and sends the resulting evaluation share to $\mathbf{HN}$. The hardened node can then choose a subset $\mathcal{S}$ of [3], combine the corresponding evaluation shares to a full random value ϕ, verify that the non-hardened devices in $\mathcal{S}$ behaved honestly, and then use ϕ as input to the key generation algorithm of the ECDSA signature scheme. Note that this key generation is deterministic, since we explicitly give the randomness ϕ as input.

The Final Derivation Mechanism While the above solution is compatible with BIP32, it has the significant drawback that each non-hardened device must maintain two secret key shares, one for the signature scheme and one for the TVRF. As a consequence, each device requires double the storage space which is an issue for space restricted devices. There is however another, more severe issue with the above solution. Similar to the signing keys, the TVRF keys must be deterministically derived throughout the wallet tree via executions of a communication heavy DPSS scheme. This incurs a significant communication overhead, especially since *all* non-hardened nodes must derive TVRF keys irrespectively of whether they want to derive a hardened node or not.

We observe that both, the DDH-based TVRF scheme of [GLOW21] (which we denote by TVRF and recall in Appendix A.2) and the ECDSA signature scheme, operate over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q and use secret/public key pairs $\mathsf{sk} \overset{\$}{\leftarrow} \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$. The security of TVRF relies on the assumption that DDH is hard in $\mathbb{G}$. Bitcoin, Ethereum and several other cryptocurrencies use the group $\mathbb{G}$ identified by the elliptic curve secp256k1, for which dlog and DDH are assumed to be hard. Therefore, our idea to mitigate the above issues is to use only a single key pair for both schemes. This allows non-hardened wallets to re-use their signing secret key shares for the TVRF during the hardened node derivation, thereby avoiding the overhead of maintaining a second key pair per wallet.

In the remainder of this section, we define a cryptographic scheme that consists of the joint procedures of the rGG scheme from Section 3 and of the DDH-based TVRF scheme, but that uses the same key pair for all procedures. We then define security properties and prove the scheme secure w.r.t. these properties.

4.3 Joint Threshold Signature/TVRF Scheme

We define a scheme (t, n)-$\mathsf{TVRF\text{-}rGG}[\mathsf{H}_0, \mathsf{H}_1]$, which consists of all procedures of the interactive (t, n)-threshold ECDSA scheme with rerandomizable keys $\mathsf{rGG}[\mathsf{H}_0]$ and the non-interactive (t, n)-threshold verifiable random function scheme $\mathsf{TVRF}[\mathsf{H}_1]$, except that it uses only one of $\mathsf{rGG}[\mathsf{H}_0].\mathsf{Gen}$ and $\mathsf{TVRF}[\mathsf{H}_1].\mathsf{Gen}$. Concretely, $\mathsf{TVRF\text{-}rGG}[\mathsf{H}_0, \mathsf{H}_1]$ consists of the procedures

$$\begin{aligned}\mathsf{TVRF\text{-}rGG}[\mathsf{H}_0, \mathsf{H}_1] =&(\mathsf{rGG}[\mathsf{H}_0].\mathsf{Gen}, \mathsf{rGG}[\mathsf{H}_0].\mathsf{RandSK}, \mathsf{rGG}[\mathsf{H}_0].\mathsf{RandPK}, \mathsf{rGG}[\mathsf{H}_0].\mathsf{TSign},\\ &\mathsf{rGG}[\mathsf{H}_0].\mathsf{Verify}, \mathsf{TVRF}[\mathsf{H}_1].\mathsf{PEval}, \mathsf{TVRF}[\mathsf{H}_1].\mathsf{Combine}, \mathsf{TVRF}[\mathsf{H}_1].\mathsf{Verify}).\end{aligned}$$

For simplicity, we sometimes abbreviate the schemes $\mathsf{TVRF\text{-}rGG}[\mathsf{H}_0, \mathsf{H}_1]$, $\mathsf{rGG}[\mathsf{H}_0]$ and $\mathsf{TVRF}[\mathsf{H}_1]$ by $\mathsf{TVRF\text{-}rGG}$, rGG and TVRF respectively. The $\mathsf{TVRF\text{-}rGG}$ scheme must satisfy the security properties *pseudorandomness*, *uniqueness*, and *robustness*. These security notions essentially combine the respective security properties of the TVRF scheme with the one-more unforgeability notion of our rGG scheme. That is, for each of the above security notions, we define a game, where an adversary (1) can corrupt t parties, (2) receives oracle access to all oracles of the one-more unforgeability game (i.e., **th-ufcma-hrk1**) and all oracles of the respective TVRF property (e.g., pseudorandomness), and (3) can win the game by either breaking the one-more unforgeability of rGG (**Case 1**) or the TVRF property (**Case 2**).

4.3.1 Pseudorandomness of TVRF-rGG

In the following we define the pseudorandomness property of $\mathsf{TVRF\text{-}rGG}$ via a game **unf-prand** and prove that $\mathsf{TVRF\text{-}rGG}$ satisfies this property. Later in Section 4.3.2, we provide the uniqueness and robustness definitions and argue that the $\mathsf{TVRF\text{-}rGG}$ scheme satisfies them.

Definition 4.1 (Pseudorandomness of $\mathsf{TVRF\text{-}rGG}$). The (t, n)-$\mathsf{TVRF\text{-}rGG}$ scheme is **unf-prand**-secure if no PPT adversary $\mathcal{A}$ wins game **unf-prand** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game **unf-prand** as

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}} &:= \Pr[\mathbf{unf\text{-}prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1|\mathbf{Case\ 1}] \cdot \Pr[\mathbf{Case\ 1}]\\ &+ \left(\Pr[\mathbf{unf\text{-}prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1|\mathbf{Case\ 2}] - \frac{1}{2}\right) \cdot \Pr[\mathbf{Case\ 2}],\end{aligned}$$

where $\Pr[\mathbf{Case\ 1}]$ and $\Pr[\mathbf{Case\ 2}]$ denote the probabilities that $\mathcal{A}$ tries to win game **unf-prand** via **Case 1** or **Case 2** respectively.

Game $\mathbf{unf\text{-}prand}_{\mathsf{TVRF\text{-}rGG}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \leq t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.
- The game initializes $\mathsf{SigList} \leftarrow \{\epsilon\}$, $\mathsf{RList} \leftarrow \{\epsilon\}$ and $\mathsf{EvalList} \leftarrow \{\epsilon\}$ and executes $(\mathsf{pk}, \{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}) \leftarrow \mathsf{TVRF\text{-}rGG}.\mathsf{Gen}(1^\kappa, t, n)$. $\mathcal{A}$ receives as input pk and $\{\mathsf{sk}_i\}_{i\in\mathsf{C}}$.
- The adversary obtains access to the following oracles:
  - `Rand`: Same as in game $\mathbf{th\text{-}ufcma\text{-}hrk1}_{\mathsf{rGG}}$.
  - `RSign`: Same as in game $\mathbf{th\text{-}ufcma\text{-}hrk1}_{\mathsf{rGG}}$.
  - `REval`: On input message m, index $i \in [n] \setminus \mathsf{C}$ and randomness ρ, check if $\rho \in \mathsf{RList}$ and abort otherwise. The oracle executes

$$\begin{aligned}(\mathsf{pk}', \mathsf{sk}'_i) \leftarrow&(\mathsf{TVRF\text{-}rGG}.\mathsf{RandPK}(\mathsf{pk}, \rho), \mathsf{TVRF\text{-}rGG}.\mathsf{RandSK}(i, \mathsf{sk}_i, \rho)),\\ (\phi_i, \pi_i) \leftarrow&\mathsf{TVRF\text{-}rGG}.\mathsf{PEval}(m, \mathsf{sk}'_i, \mathsf{pk}')\end{aligned}$$

    and if $(i, m, \rho) \notin \mathsf{EvalList}$, stores the tuple (i, m, ρ) in $\mathsf{EvalList}$. The oracle returns (ϕ_i, π_i).
- The adversary wins the game if it wins either of the following cases:

– **Case 1**: Same as in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$.

– **Case 2**: The adversary outputs a message m^*, a randomness ρ^*, a set of indices $\mathcal{S}$ with $|\mathcal{S}| > t$ and evaluation shares $\{\phi_k, \pi_k\}_{k\in\mathcal{S}\cap\mathsf{C}}$. The game checks if there are less than $t - |\mathcal{S} \cap \mathsf{C}|$ tuples of the form $(\cdot, m^*, \rho^*)$ in $\mathsf{EvalList}$ and if $\rho^* \in \mathsf{RList}$. If so, for $i \in \mathcal{S} \setminus \mathsf{C}$ the game computes

$$\begin{aligned}(\mathsf{pk}', \mathsf{sk}'_i) &\leftarrow (\mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho^*), \mathsf{TVRF\text{-}rGG.RandSK}(i, \mathsf{sk}_i, \rho^*)),\\ (\phi_i, \pi_i) &\leftarrow \mathsf{TVRF\text{-}rGG.PEval}(m^*, \mathsf{sk}'_i, \mathsf{pk}')\\ (\phi, \pi) &\leftarrow \mathsf{TVRF\text{-}rGG.Combine}(\mathsf{pk}', \mathcal{S}, \{(\phi_j, \pi_j)\}_{j\in\mathcal{S}}).\end{aligned}$$

If $\phi = \perp$, the game returns ϕ. Otherwise the game chooses a bit $b \xleftarrow{\$} \{0, 1\}$ and does the following:

∗ If $b = 0$: Return ϕ.

∗ If $b = 1$: Sample $\phi' \xleftarrow{\$} \mathbb{G}$ and output ϕ'.

The adversary then outputs a bit b' and wins if $b = b'$.

Theorem 4.2 *Let $\mathsf{H}_0 : \{0, 1\}^* \to \mathbb{Z}_q$, $\mathsf{H}_1 : \{0, 1\}^* \to \mathbb{G}$ be hash functions modeled as a random oracle. Let $\mathsf{rGG}[\mathsf{H}_0]$ be the* **th-ufcma-hrk1***-secure interactive (t, n)-threshold ECDSA scheme with rerandomizable keys from Section 3.2 and let $\mathsf{TVRF}[\mathsf{H}_1]$ be the* **th-prand***-secure (t, n)-threshold verifiable random function as described in Appendix A.2. Further, let* PKE *be a semantically secure linearly homomorphic encryption scheme,* ZK *and* DLEq *as described in Appendix A.2 be non-interactive zero-knowledge proof systems,* CT *a non-malleable and equivocable commitment scheme and the DDH assumption hold in $\mathbb{G}$. Then the (t, n)-$\mathsf{TVRF\text{-}rGG}[\mathsf{H}_0, \mathsf{H}_1]$ scheme as described above is* **unf-prand***-secure.*

Proof. In order to prove this theorem we provide a reduction either to the one-more unforgeability of the rGG scheme, i.e., to the **th-ufcma-hrk1** security of rGG or to the pseudorandomness property of the TVRF scheme, i.e., to the **th-prand** security of TVRF. In other words, we show that if a PPT adversary $\mathcal{A}$ is able to win the $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ game with more than negligible advantage, then we can construct a PPT adversary $\mathcal{B}$ which can either win the $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ or the $\textbf{th-prand}_{\mathsf{TVRF}}$ game with more than negligible advantage. To this end, $\mathcal{B}$ first guesses if $\mathcal{A}$ is going to win game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ through **Case 1** or **Case 2**. Depending on the guess, $\mathcal{B}$ decides to either play in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ or $\textbf{th-prand}_{\mathsf{TVRF}}$, while simulating $\mathcal{A}$'s oracle queries. Note that $\mathcal{B}$ receives only access to the oracles of either game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ *or* $\textbf{th-prand}_{\mathsf{TVRF}}$ which significantly complicates the simulation of $\mathcal{A}$'s oracle queries. In particular, when playing in game $\textbf{th-prand}_{\mathsf{TVRF}}$, $\mathcal{B}$ does not get access to a signing oracle, yet has to simulate oracle `RSign` of game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ to $\mathcal{A}$.

Let $\mathcal{B} := (\mathcal{B}_0, \mathcal{B}_1)$ be composed of two subprocedures. At the beginning of game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$, $\mathcal{B}$ chooses a bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 0$, $\mathcal{B}$ executes subprocedure $\mathcal{B}_0$ that plays in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ and otherwise $\mathcal{B}$ executes $\mathcal{B}_1$ that plays in game $\textbf{th-prand}_{\mathsf{TVRF}}$. In the following, we show for both cases (i.e., $b = 0$ and $b = 1$) that the respective subprocedure $\mathcal{B}_b$ can simulate game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ to $\mathcal{A}$ and use $\mathcal{A}$'s output to win their respective security games (i.e., either $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ or $\textbf{th-prand}_{\mathsf{TVRF}}$). Finally, after analyzing both cases separately, we determine the advantage of $\mathcal{B} := (\mathcal{B}_0, \mathcal{B}_1)$ to win either game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ or game $\textbf{th-prand}_{\mathsf{TVRF}}$. We additionally provide an intuitive proof sketch for both cases in Appendix B.3.

Case $b = 0$ In this case we show via a series of computationally indistinguishable games that there exists an adversary $\mathcal{B}_0$ which can use adversary $\mathcal{A}$ in **Case 1** to win its own game $\textbf{th-ufcma-hrk1}^{\mathcal{B}_0}_{\mathsf{rGG}}$. Game $\boldsymbol{G}_0$: This game is the original $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ game, in which adversary $\mathcal{A}$ can corrupt t parties and receives access to oracles H_0, H_1, `RSign`, `Rand` and `REval`. We have $\Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1|\textbf{Case 1}] = \Pr[\boldsymbol{G}^{\mathcal{A}}_0 = 1]$.

Game $\boldsymbol{G}_1$: This game is similar to the previous game with two differences. First, in the beginning the game initializes a set $\mathsf{HList} := \epsilon$. Second, upon $\mathcal{A}$ sending a query to H_1 on input m, if $\mathsf{H}_1(m) = \perp$ the game samples uniformly at random $r \xleftarrow{\$} \mathbb{Z}_q$, sets $\mathsf{HList}[m] := r$ and $\mathsf{H}_1(m) := g^r$. The game outputs $\mathsf{H}_1(m)$.

It is easy to see that the random oracle H_1 returns uniformly random group elements since r is chosen uniformly at random from $\mathbb{Z}_q$. Therefore, we have that $\Pr[\boldsymbol{G}^{\mathcal{A}}_1 = 1] = \Pr[\boldsymbol{G}^{\mathcal{A}}_0 = 1]$.

Game $\boldsymbol{G}_2$: This game is similar to the previous game with a difference in the REval oracle. On input a message m, an index i and a randomness ρ, the game executes

$$\begin{aligned}\mathsf{pk}' &\leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho),\\ \mathsf{sk}_i' &\leftarrow \mathsf{TVRF\text{-}rGG.RandSK}(i, \mathsf{sk}_i, \rho),\\ (\phi_i, \pi_i) &\leftarrow \mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}_i', \mathsf{pk}').\end{aligned}$$

However, instead of outputting (ϕ_i, π_i), the game simulates a zero-knowledge proof π_i' that proves correctness of ϕ_i and outputs (ϕ_i, π_i').

Due to the zero-knowledge property of the proof system DLEq (cf. Appendix A.2), this game is indistinguishable from the previous one except with negligible probability. That is, we have that $\Pr[\boldsymbol{G}_1^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] + \mathsf{negl}(\kappa)$ where negl is a negligible function in the security parameter κ.

Game $\boldsymbol{G}_3$: This game is similar to the previous game with a difference in the REval oracle. On input a message m, an index i, and a randomness ρ, the game checks if $\mathsf{HList}[m] = \perp$. If so, it queries $\mathsf{H}_1(m)$. It then executes $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho)$, parses $\mathsf{pk}' := (X', \{X_1', \cdots, X_n'\})$ and computes $\phi_i \leftarrow (X_i')^r$ for $r \leftarrow \mathsf{HList}[m]$.

This game is equivalent to the previous game since $(X_i')^r = g^{\mathsf{sk}_i' \cdot r} = \mathsf{H}_1(m)^{\mathsf{sk}_i'}$. Therefore, we have that $\Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_3^{\mathcal{A}} = 1]$.

Reduction to **th-ufcma-hrk1**$_{\mathsf{rGG}}$: Having shown that the transition from game $\boldsymbol{G}_0$ to game $\boldsymbol{G}_3$ is indistinguishable, it remains to show that an adversary $\mathcal{A}$ winning in game $\boldsymbol{G}_3$ can be used to construct an adversary $\mathcal{B}_0$ that wins game **th-ufcma-hrk1**$_{\mathsf{rGG}}$. To do so, we must show that $\mathcal{B}_0$ playing in **th-ufcma-hrk1**$_{\mathsf{rGG}}$ can simulate game $\boldsymbol{G}_3$ to $\mathcal{A}$. The simulation differs from game $\boldsymbol{G}_3$ in the following ways:

1. $\mathcal{B}_0$ does not generate the secret key shares and public key, but instead corrupts the same set of parties C in **th-ufcma-hrk1**$_{\mathsf{rGG}}$ as $\mathcal{A}$ does in $\boldsymbol{G}_3$. $\mathcal{B}_0$ then forwards the public key pk and the secret key shares $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$ from game **th-ufcma-hrk1**$_{\mathsf{rGG}}$ to $\mathcal{A}$.

2. Upon $\mathcal{A}$ querying oracle RSign on input a message m and a randomness ρ, $\mathcal{B}_0$ queries its own oracle RSign on input m and ρ and relays the messages between $\mathcal{A}$ and the RSign oracle in game **th-ufcma-hrk1**$_{\mathsf{rGG}}$.

3. Upon $\mathcal{A}$ querying oracle H_0 on input a message m, $\mathcal{B}_0$ forwards the query to its own random oracle and relays the output.

It is easy to see that $\mathcal{B}_0$'s simulation is indistinguishable from game $\boldsymbol{G}_3$ to $\mathcal{A}$. It remains to show that $\mathcal{B}_0$ can use $\mathcal{A}$'s forgery in game **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$ to win its own game **th-ufcma-hrk1**$_{\mathsf{rGG}}$. Since $\mathcal{B}_0$ forwards all queries to RSign in game **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$ to the corresponding oracle in game **th-ufcma-hrk1**$_{\mathsf{rGG}}$, $\mathcal{B}_0$ and $\mathcal{A}$ query their respective oracles on the same messages. Therefore, if $\mathcal{A}$ outputs a valid forgery in **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$, then the forgery is also valid in **th-ufcma-hrk1**$_{\mathsf{rGG}}$. We finally have

$$\begin{aligned}\Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1 | \textbf{Case 1}] &= \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_3^{\mathcal{A}} = 1] + \mathsf{negl}(\kappa)\\ &= \Pr[\textbf{th-ufcma-hrk1}^{\mathcal{B}_0}_{\mathsf{rGG}[\mathsf{H}_0]} = 1] + \mathsf{negl}(\kappa)\\ &= \mathsf{Adv}^{\mathcal{B}_0}_{\textbf{th-ufcma-hrk1}_{\mathsf{rGG}[\mathsf{H}_0]}} + \mathsf{negl}(\kappa).\end{aligned}$$

Case $b = 1$ We now show via a series of computationally indistinguishable games that there exists an adversary $\mathcal{B}_1$ which can use adversary $\mathcal{A}$ in **Case 2** to win its own game **th-prand**$^{\mathcal{B}_1}_{\mathsf{TVRF}}$.

Game $\boldsymbol{G}_0$: This game is the original **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$ game, in which adversary $\mathcal{A}$ can corrupt t parties and receives access to oracles H_0, H_1, RSign, Rand and REval. We have $\Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1 | \textbf{Case 2}] = \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_1$: This game is similar to the previous game with two differences. First, in the beginning the game initializes a set $\mathsf{HSigs} := \epsilon$. Second, upon a query to H_0 on input a public key prefixed message

$m := (\mathsf{pk}', m')$, i.e., where $(\mathsf{pk}', \cdot) \in \mathsf{TVRF\text{-}rGG.Gen}(1^\kappa, t, n)$, the game checks if $\mathsf{H}_0(m) = \bot$. If so, it executes $\mathcal{S}_{\mathsf{ECDSA}}$ as described in Figure 3 on input (X', m) where $\mathsf{pk}' := (X', \{X'_1, \cdots, X'_n\})$. Finally, the game sets $\mathsf{HSigs}[m] := (r, s)$.

$$\boxed{\begin{array}{l}\mathcal{S}_{\mathsf{ECDSA}}(X, m):\\ a, b \xleftarrow{\$} \mathbb{Z}_q,\ R = X^a \cdot g^b,\ r = f(R),\ s = \frac{r}{a},\ \mathsf{H}_0(m) := \frac{r \cdot b}{a}\end{array}}$$

Figure 3: Simulation of ECDSA signatures via programming of the random oracle H_0 as first presented by Fersch et al. [FKP17]. The function $f : \mathbb{G} \to \mathbb{Z}_q$ is defined as the projection of a group element to its x-coordinate.

It is easy to see that $\mathcal{S}_{\mathsf{ECDSA}}$ programs the random oracle H_0 in such a way that H_0 returns uniformly random values. Therefore, this game is equivalent to the previous game, i.e., $\Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_1^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_2$: This game is similar to the previous game with a difference in the `RSign` oracle. On input a message m and a randomness ρ, the game first computes the rerandomized (full) secret key sk' and then generates a full ECDSA signature σ' under sk' for message m. The game then executes the signing procedure in the same way as presented in the proof sketch of Theorem 3.4 using signature σ'.

The indistinguishability of this game to the previous one follows in the same way as in Theorem 3.4. Note that the simulation of the signing procedure as described in the proof sketch of Theorem 3.4 does not program H_0 and therefore does not conflict with the execution of $\mathcal{S}_{\mathsf{ECDSA}}$. We have that $\Pr[\boldsymbol{G}_1^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] + \mathsf{negl}_1(\kappa)$ where negl_1 is a negligible function in the security parameter κ.

Game $\boldsymbol{G}_3$: This game is similar to the previous game again with a modification in the `RSign` oracle. On input a message m and a randomness ρ, the game does not generate a full ECDSA signature using sk', but it fetches $(r, s) \leftarrow \mathsf{HSigs}[m']$ for $m' \leftarrow (\mathsf{pk}', m)$ where $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho)$.[8] The game then uses the tuple (r, s) as the full ECDSA signature under sk'.

As shown in [FKP17], the tuple (r, s) as generated by the $\mathcal{S}_{\mathsf{ECDSA}}$ algorithm (see Figure 3) is computationally indistinguishable from an honestly generated ECDSA signature for message m' under public key X' (where $\mathsf{pk}' := (X', \{X'_1, \cdots, X'_n\})$) to a PPT adversary $\mathcal{A}$ with access to random oracle H_0. Since the simulation of the signing procedure as described in the proof sketch of Theorem 3.4 forces the execution of the `RSign` oracle to output (r, s), adversary $\mathcal{A}$ can distinguish this game from the previous one only with negligible probability. Therefore, we have that $\Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_3^{\mathcal{A}} = 1] + \mathsf{negl}_2(\kappa)$ where negl_2 is a negligible function in the security parameter κ.

Game $\boldsymbol{G}_4$: This game is similar to the previous game with a modification in the `REval` oracle. On input a message m, an index i and a randomness ρ, the game computes $\mathsf{sk}'_i \leftarrow \mathsf{TVRF\text{-}rGG.RandSK}(i, \mathsf{sk}_i, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho)$ and executes $(\phi_i, \pi_i) \leftarrow \mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}'_i, \mathsf{pk}')$. Instead of outputting (ϕ_i, π_i), however, the game simulates a zero-knowledge proof π'_i that proves correctness of ϕ_i. The game then outputs (ϕ_i, π'_i).

Due to the zero-knowledge property of the proof system DLEq (cf. Appendix A.2), this game is indistinguishable from the previous one except with negligible probability. Therefore, we have that $\Pr[\boldsymbol{G}_3^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_4^{\mathcal{A}} = 1] + \mathsf{negl}_3(\kappa)$ where negl_3 is a negligible function in the security parameter κ.

Game $\boldsymbol{G}_5$: This game is similar to the previous game with a modification in the `REval` oracle. On input a message m, an index i, and a randomness ρ, instead of rerandomizing the secret key share sk_i to sk'_i and executing $\mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}'_i, \mathsf{pk}')$, the game computes

$$\mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}_i, \mathsf{pk}) \cdot \mathsf{H}_1(m)^{\rho_i},$$

where ρ_i denotes the randomness share of ρ for party P_i according to the $\mathsf{TVRF\text{-}rGG.RandSK}$ (cf. Figure 1) algorithm.

This game is equivalent to the previous game since for $\mathsf{sk}'_i \leftarrow \mathsf{TVRF\text{-}rGG.RandSK}(i, \mathsf{sk}_i, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho)$ it holds that:

$$\mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}'_i, \mathsf{pk}') = \mathsf{TVRF\text{-}rGG.PEval}(m, \mathsf{sk}_i, \mathsf{pk}) \cdot \mathsf{H}_1(m)^{\rho_i} = \mathsf{H}_1(m)^{\mathsf{sk}_i + \rho_i}.$$

[8]We assume that H_0 has been queried on m' before the signing query.

Therefore, we have that $\Pr[\boldsymbol{G}_4^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1]$.

Game $\boldsymbol{G}_6$: This game is similar to the previous game with a modification in the challenge phase. Upon $\mathcal{A}$ outputting a message m^*, randomness ρ^*, a set of indices $\mathcal{S}$, and evaluation shares $\{(\phi_k^*, \pi_k^*)\}_{k \in \mathcal{S} \cap \mathsf{C}}$, the game verifies the proofs π_k^* and returns $\perp$ if any proof does not verify. Otherwise the game checks if $\phi_k^* = \mathsf{H}_1(m^*)^{\mathsf{sk}_k + \rho_k^*}$ and aborts if any of these checks does not hold.

Note that the only way that the game aborts is if the adversary manages to submit a verifying zero-knowledge proof π_k^* for a false statement. Due to the soundness property of the DLEq proof system, this event happens only with negligible probability. Therefore, we have that $\Pr[\boldsymbol{G}_5^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] + \mathsf{negl}_4(\kappa)$ where negl_4 is a negligible function in the security parameter κ.

Game $\boldsymbol{G}_7$: This game is similar to the previous game with a modification in the challenge phase. Namely, upon $\mathcal{A}$ outputting a message m^*, randomness ρ^*, a set of indices $\mathcal{S}$, and evaluation shares $\{(\phi_k^*, \pi_k^*)\}_{k \in \mathcal{S} \cap \mathsf{C}}$ the game computes $\phi_k = \phi_k^* \cdot \mathsf{H}_1(m^*)^{-\rho_k^*} = \mathsf{H}_1(m^*)^{\mathsf{sk}_k}$ and generates a new zero-knowledge proof π_k using sk_k and ϕ_k. The game then computes $(\phi_i, \pi_i) \leftarrow \mathsf{TVRF\text{-}rGG.PEval}(m^*, \mathsf{sk}_i, \mathsf{pk})$ for $i \in \mathcal{S} \setminus \mathsf{C}$ and $(\phi, \pi) \leftarrow \mathsf{TVRF\text{-}rGG.Combine}(\mathsf{pk}, \mathcal{S}, \{(\phi_j, \pi_j)\}_{j \in \mathcal{S}})$. The game chooses uniformly at random $b \xleftarrow{\$} \{0, 1\}$ and if $b = 0$ sets $\phi' := \phi$ and otherwise sets $\phi' \xleftarrow{\$} \mathbb{G}$. Finally, the game computes $\phi^* = \phi' \cdot \mathsf{H}_1(m^*)^{\rho^*}$ and returns it to $\mathcal{A}$.

Note that if ϕ' was chosen randomly from $\mathbb{G}$ by the game then ϕ^* is also a uniformly random element, and if ϕ' is a valid TVRF output, then so is ϕ^* under the key randomized with ρ^*. This is since $\phi^* = \phi' \cdot \mathsf{H}_1(m^*)^{\rho^*} = \mathsf{H}_1(m^*)^{\mathsf{sk}} \cdot \mathsf{H}_1(m^*)^{\rho^*} = \mathsf{H}_1(m^*)^{\mathsf{sk}+\rho^*}$. We have that $\Pr[\boldsymbol{G}_6^{\mathcal{A}} = 1] = \Pr[\boldsymbol{G}_7^{\mathcal{A}} = 1]$.

Reduction to $\textbf{th-prand}_{\mathsf{TVRF}}$: Having shown that the transition from game $\boldsymbol{G}_0$ to game $\boldsymbol{G}_7$ is indistinguishable, it remains to show that an adversary $\mathcal{A}$ winning in game $\boldsymbol{G}_7$ can be used to construct an adversary $\mathcal{B}_1$ that wins game $\textbf{th-prand}_{\mathsf{TVRF}}$. To do so, we must show that $\mathcal{B}_1$ playing in $\textbf{th-prand}_{\mathsf{TVRF}}$ can simulate game $\boldsymbol{G}_7$ to $\mathcal{A}$. The simulation differs from game $\boldsymbol{G}_7$ in the following points:

1. $\mathcal{B}_1$ does not generate the secret key shares and public key, but instead corrupts the same set of parties C in $\textbf{th-prand}_{\mathsf{TVRF}}$ as $\mathcal{A}$ does in $\boldsymbol{G}_7$. $\mathcal{B}_1$ then forwards the public key pk and the secret key shares $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$ from game $\textbf{th-prand}_{\mathsf{TVRF}}$ to $\mathcal{A}$.

2. Upon $\mathcal{A}$ querying oracle `REval` on input a message m, an index i and a randomness ρ, $\mathcal{B}_1$ queries its own oracle `Eval` on input m and i and uses the oracle output to compute the output of `REval` as in $\boldsymbol{G}_7$.

3. Upon $\mathcal{A}$ querying oracle H_1 on input a message m, $\mathcal{B}_1$ forwards the query to its own random oracle and relays the output.

4. During the challenge phase, $\mathcal{B}_1$ sends the shares $\phi_k = \phi_k^* \cdot \mathsf{H}_1(m^*)^{-\rho_k^*}$ together with the zero-knowledge proofs π_k to its own game and receives an element ϕ^*. $\mathcal{B}_1$ forwards to $\mathcal{A}$ the element $\phi^* \cdot \mathsf{H}_1(m^*)^{\rho^*}$.

It is easy to see that $\mathcal{B}_1$'s simulation is indistinguishable from game $\boldsymbol{G}_7$ to $\mathcal{A}$ and that if $\mathcal{A}$ wins game $\boldsymbol{G}_7$ with more than negligible probability, then $\mathcal{B}_1$ wins game $\textbf{th-prand}_{\mathsf{TVRF}}$ with the same probability. The latter is because $\mathcal{B}_1$ makes the same queries to oracle `Eval` as $\mathcal{A}$ does to oracle `REval`. We finally have that

$$\begin{aligned}\Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1 | \textbf{Case 2}] &= \Pr[\boldsymbol{G}_0^{\mathcal{A}} = 1] \leq \Pr[\boldsymbol{G}_7^{\mathcal{A}} = 1] + \mathsf{negl}'(\kappa) \\ &= \Pr[\textbf{th-prand}^{\mathcal{B}_1}_{\mathsf{TVRF}[\mathsf{H}_1]} = 1] + \mathsf{negl}'(\kappa),\end{aligned}$$

where $\mathsf{negl}'(\kappa) := \sum_{i=1}^{4} \mathsf{negl}_i(\kappa)$.

Finally, we determine the advantage of adversary $\mathcal{B} := (\mathcal{B}_0, \mathcal{B}_1)$ to win either in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ or $\textbf{th-prand}_{\mathsf{TVRF}}$. Note that $\mathcal{B}$'s advantage is:

$$\mathsf{Adv}^{\mathcal{B}} := \frac{1}{2}\mathsf{Adv}^{\mathcal{B}_0}_{\textbf{uf-cma-hrk1}_{\mathsf{rECDSA}[\mathsf{H}_0]}} \cdot \Pr[\textbf{Case 1}] + \frac{1}{2}\mathsf{Adv}^{\mathcal{B}_1}_{\textbf{th-prand}_{\mathsf{TVRF}[\mathsf{H}_1]}} \cdot \Pr[\textbf{Case 2}].$$

Therefore we can conclude that:

$$\begin{aligned}
\mathsf{Adv}^{\mathcal{A}} &:= \Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1 | \textbf{Case 1}] \cdot \Pr[\textbf{Case 1}] \\
&\quad + (\Pr[\textbf{unf-prand}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1 | \textbf{Case 2}] - \frac{1}{2}) \cdot \Pr[\textbf{Case 2}] \\
&\leq (\Pr[\textbf{th-ufcma-hrk1}^{\mathcal{B}_0}_{\mathsf{rGG[H_0]}} = 1] + \mathsf{negl}(\kappa)) \cdot \Pr[\textbf{Case 1}] \\
&\quad + (\Pr[\textbf{th-prand}^{\mathcal{B}_1}_{\mathsf{TVRF[H_1]}} = 1] + \mathsf{negl}'(\kappa) - \frac{1}{2}) \cdot \Pr[\textbf{Case 2}] \\
&\leq (\mathsf{Adv}^{\mathcal{B}_0}_{\textbf{uf-cma-hrk1}_{\mathsf{rECDSA[H_0]}}} + \mathsf{negl}(\kappa)) \cdot \Pr[\textbf{Case 1}] \\
&\quad + (\mathsf{Adv}^{\mathcal{B}_1}_{\textbf{th-prand}_{\mathsf{TVRF[H_1]}}} + \mathsf{negl}'(\kappa)) \cdot \Pr[\textbf{Case 2}] \\
&= 2 \cdot \mathsf{Adv}^{\mathcal{B}} + \mathsf{negl}''(\kappa)
\end{aligned}$$

where $\mathsf{negl}''(\kappa) := \mathsf{negl}(\kappa) \cdot \Pr[\textbf{Case 1}] + \mathsf{negl}'(\kappa) \cdot \Pr[\textbf{Case 2}]$ is a negligible function in κ. ■

4.3.2 Uniqueness and Robustness of TVRF-rGG

Besides pseudorandomness, the TVRF-rGG scheme must additionally satisfy the properties of uniqueness and robustness, which are defined in a similar manner as the pseudorandomness property in the sense that they combine the respective property of the TVRF scheme with the one-more unforgeability of our rGG scheme. In the following we provide the formal definitions of these two properties.

Definition 4.3 (Uniqueness of TVRF-rGG). The (t, n)-TVRF-rGG scheme is unique if no PPT adversary $\mathcal{A}$ wins game **unf-unique** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{unf-unique}_{\mathsf{TVRF\text{-}rGG}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{unf-unique}} := \Pr[\textbf{unf-unique}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1]$.

Game $\textbf{unf-unique}_{\mathsf{TVRF\text{-}rGG}}$:

- The adversary $\mathcal{A}$ outputs a list of corrupted parties C, such that $|\mathsf{C}| \leq t$ and for all $i \in \mathsf{C}$ it holds that $i \in [n]$.
- The game initializes $\mathsf{SigList} \leftarrow \{\epsilon\}$ and $\mathsf{RList} \leftarrow \{\epsilon\}$ and executes $(\mathsf{pk}, \{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\}) \leftarrow \mathsf{TVRF\text{-}rGG.Gen}(1^\kappa, t, n)$. Then $\mathcal{A}$ is run on input pk and $\{\mathsf{sk}_i\}_{i \in \mathsf{C}}$.
- The adversary obtains access to the following oracles:
  - `Rand`: Same as in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$.
  - `RSign`: Same as in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$.
  - `REval`: Same as in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$.
  - `KeyLeak`: On input $i \in [n]$, the oracle outputs sk_i.
- The adversary wins the game if it wins either of the following cases:
  - **Case 1**: Output 0 if there has been any query to oracle `KeyLeak`. Otherwise this case is the same as **Case 1** in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$.
  - **Case 2**: The adversary outputs a message m^*, a randomness ρ^* and evaluations $\{(\phi^{i^*}, \pi^{i^*})\}_{i \in \{0,1\}}$. If $\rho^* \in \mathsf{RList}$, the game computes $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho^*)$. The game outputs 1 if $\phi^{0^*} \neq \phi^{1^*}$ and

$$\mathsf{TVRF\text{-}rGG.Verify}(\mathsf{pk}', m^*, \phi^{0^*}, \pi^{0^*}) = \mathsf{TVRF\text{-}rGG.Verify}(\mathsf{pk}', m^*, \phi^{1^*}, \pi^{1^*}) = 1.$$

    Otherwise it outputs 0.

Definition 4.4 (Robustness of TVRF-rGG). The (t, n)-TVRF-rGG scheme is robust if no PPT adversary $\mathcal{A}$ wins game **unf-robust** as described below with more than negligible advantage. We define $\mathcal{A}$'s advantage in game $\textbf{unf-robust}_{\mathsf{TVRF\text{-}rGG}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{unf-robust}} := \Pr[\textbf{unf-robust}^{\mathcal{A}}_{\mathsf{TVRF\text{-}rGG}} = 1]$.

Game **unf-robust**$_{\mathsf{TVRF\text{-}rGG}}$:

- The game is exactly the same as game **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$, except for the winning conditions, which we will describe below.
- The adversary wins the game if it wins either of the following cases:
  - **Case 1**: Same as **Case 1** in game **unf-prand**$_{\mathsf{TVRF\text{-}rGG}}$.
  - **Case 2**: The adversary outputs a message m^*, a set $\mathcal{S}$ with $|\mathcal{S}| > t$, a list of evaluation shares $\{\phi_i, \pi_i\}_{i \in \mathcal{S} \cap \mathsf{C}}$ and a randomness ρ^*. The game checks if $\rho^* \in \mathsf{RList}$ and if so computes for all $i \in \mathcal{S} \setminus \mathsf{C}$:

$$\begin{aligned}(\mathsf{pk}', \mathsf{sk}'_i) \leftarrow &(\mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho^*), \mathsf{TVRF\text{-}rGG.RandSK}(i, \mathsf{sk}_i, \rho^*)),\\ (\phi_i, \pi_i) \leftarrow &\mathsf{TVRF\text{-}rGG.PEval}(m^*, \mathsf{sk}'_i, \mathsf{pk}')\end{aligned}$$

    The game finally sets

$$(\phi^*, \pi^*) \leftarrow \mathsf{TVRF\text{-}rGG.Combine}(\mathsf{pk}', \mathcal{S}, \{\phi_i, \pi_i\}_{i \in \mathcal{S}}).$$

    If $\phi^* \neq \perp$ and $\mathsf{TVRF\text{-}rGG.Verify}(\mathsf{pk}', m^*, \phi^*, \pi^*) = 0$, the game outputs 1 and 0 otherwise.

The proof of the uniqueness and robustness property of the $\mathsf{TVRF\text{-}rGG}$ scheme is similar to the proof of the pseudorandomness property in the sense that the reduction guesses whether the adversary is going to win in **Case 1** or **Case 2**. In **Case 1**, we reduce to the one-more unforgeability of rGG. The simulation of the respective security game works in the same way as in the proof of Theorem 4.2. In **Case 2**, we can show a contradiction to the soundness property of the NIZK proof system DLEq (cf. Appendix A.2) in the same way as was previously shown by Galindo et al. [GLOW21]). The simulation of the `RSign` oracle is then straightforward since the reduction can choose the initial key set itself.

Acknowledgments

This work is supported by the German Research Foundation (DFG) Emmy Noether Program FA 1320/1-1, by the German Research Foundation DFG - SFB 1119 - 236615297 (CROSSING Projects P1 and S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain Project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE*.

References

[ADE$^+$20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. pages 1017–1031, 2020. (Cited on page 4, 8.)

[AGKK19] Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. pages 426–445, 2019. (Cited on page 1, 4.)

[AHS20] Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. A survey of ecdsa threshold signing. Cryptology ePrint Archive, Paper 2020/1390, 2020. `https://eprint.iacr.org/2020/1390`. (Cited on page 4.)

[BDLO15] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security*, page 23–41, Cham, 2015. Springer International Publishing. (Cited on page 11.)

[BMP22] Constantin Blokh, Nikolaos Makriyannis, and Udi Peled. Efficient asymmetric threshold ecdsa for mpc-based cold storage. Cryptology ePrint Archive, Paper 2022/1296, 2022. `https://eprint.iacr.org/2022/1296`. (Cited on page 4.)

[BST21] Charlotte Bonte, Nigel Smart, and Titouan Tanguy. Thresholdizing hasheddsa: Mpc to the rescue. *International Journal of Information Security*, 20, 12 2021. (Cited on page 11.)

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). pages 11–19, 1988. (Cited on page 11.)

[CCL$^+$20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. pages 266–296, 2020. (Cited on page 3, 4.)

[CCL$^+$21] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa revisited: Online/offline extensions, identifiable aborts, proactivity and adaptive security. Cryptology ePrint Archive, Paper 2021/291, 2021. https://eprint.iacr.org/2021/291. (Cited on page 3, 4.)

[CEV14] Nicolas T. Courtois, Pinar Emirdag, and Filippo Valsorda. Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor RNG events. Cryptology ePrint Archive, Report 2014/848, 2014. https://eprint.iacr.org/2014/848. (Cited on page 4.)

[CGG$^+$20a] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. pages 1769–1787, 2020. (Cited on page 3, 4.)

[CGG$^+$20b] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, New York, NY, USA, 2020. Association for Computing Machinery. (Cited on page 26.)

[CHL23] ChihYun Chuang, IHung Hsu, and TingFang Lee. A two-party hierarchical deterministic wallets in practice. In Sabrina De Capitani di Vimercati and Pierangela Samarati, editors, *Proceedings of the 20th International Conference on Security and Cryptography, SECRYPT 2023, Rome, Italy, July 10-12, 2023*, pages 850–856. SCITEPRESS, 2023. (Cited on page 3, 4.)

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. pages 89–105, 1993. (Cited on page 23.)

[DEF$^+$21] Poulami Das, Andreas Erwig, Sebastian Faust, Julian Loss, and Siavash Riahi. The exact security of bip32 wallets. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, New York, NY, USA, 2021. Association for Computing Machinery. (Cited on page 2, 3, 4, 5, 8, 9, 10, 24.)

[DFL19] Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. pages 651–668, 2019. (Cited on page 4.)

[DMZ$^+$21] Yi Deng, Shunli Ma, Xinxuan Zhang, Hailong Wang, Xuyang Song, and Xiang Xie. Promise σ-protocol: How to construct efficient threshold ecdsa from encryptions based on class groups. Springer-Verlag, 2021. (Cited on page 3, 4.)

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. pages 1–17, 2003. (Cited on page 3.)

[Ele13] Version bytes for BIP32 extended public and private keys. https://electrum.readthedocs.io/en/latest/xpub_version_bytes.html, 2013. (Cited on page 2.)

[ER22] Andreas Erwig and Siavash Riahi. Deterministic wallets for adaptor signatures. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng, editors, *Computer Security – ESORICS 2022*, pages 487–506, Cham, 2022. Springer Nature Switzerland. (Cited on page 4.)

[FKM$^+$16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. pages 301–330, 2016. (Cited on page 5.)

[FKP17] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the one-per-message unforgeability of (EC)DSA and its variants. pages 519–534, 2017. (Cited on page 16, 29.)

[GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, New York, NY, USA, 2018. Association for Computing Machinery. (Cited on page 3, 4, 9, 24, 27, 28.)

[GLOW21] David Galindo, Jia Liu, Mihair Ordean, and Jin-Mann Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 88–102, 2021. (Cited on page 6, 11, 12, 19, 23.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. pages 218–229, 1987. (Cited on page 11.)

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Cited on page 11.)

[GS15] Gus Gutoski and Douglas Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. pages 497–504, 2015. (Cited on page 4.)

[GS22] Jens Groth and Victor Shoup. Design and analysis of a distributed ecdsa signing service. Cryptology ePrint Archive, Paper 2022/506, 2022. `https://eprint.iacr.org/2022/506`. (Cited on page 4.)

[KMOS21] Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh when you wake up: Proactive threshold wallets with offline devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 608–625, 2021. (Cited on page 1, 4.)

[LFA20] Adriano Di Luzio, Danilo Francati, and Giuseppe Ateniese. Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. pages 323–343, 2020. (Cited on page 4.)

[Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. pages 613–644, 2017. (Cited on page 4.)

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1837–1854, New York, NY, USA, 2018. Association for Computing Machinery. (Cited on page 4.)

[MPs19] Antonio Marcedone, Rafael Pass, and abhi shelat. Minimizing trust in hardware wallets with two factor signatures. pages 407–425, 2019. (Cited on page 1, 4.)

[MZW$^+$19] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. Churp: Dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2369–2386, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on page 11.)

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. pages 223–238, 1999. (Cited on page 26.)

[SLL10] David Schultz, Barbara Liskov, and Moses Liskov. Mpss: Mobile proactive secret sharing. *ACM Trans. Inf. Syst. Secur.*, 13(4), dec 2010. (Cited on page 11.)

[Tre14] Trezor Wiki,Cryptocurrency standards,Hierachical deterministic wallets. https://wiki.trezor.io/Cryptocurrency_standards, 2014. (Cited on page 2.)

[Wik18] Bitcoin Wiki. BIP32 proposal. https://en.bitcoin.it/wiki/BIP_0032, 2018. (Cited on page 1.)

[Yeh23] Yehuda Lindell. Cryptography and MPC in Coinbase Wallet as a Service (WaaS). https://coinbase.bynder.com/m/687ea39fd77aa80e/original/CB-MPC-Whitepaper.pdf, 2023. (Cited on page 3.)

[YLY$^+$22] Xin Yin, Zhen Liu, Guomin Yang, Guoxing Chen, and Haojin Zhu. Secure hierarchical deterministic wallet supporting stealth address. Cryptology ePrint Archive, Paper 2022/627, 2022. https://eprint.iacr.org/2022/627. (Cited on page 4.)

A Additional Preliminaries

A.1 Correctness and one-per message unforgeability under honestly rerandomizable keys of signature schemes with rerandomizable keys

For the empty string ϵ, we have $\mathsf{RandPK}(\mathsf{pk}, \epsilon) = \mathsf{pk}$ and $\mathsf{RandSK}(\mathsf{sk}, \epsilon) = \mathsf{sk}$.

We further require:

1. *(Perfect) rerandomizability of keys:* For all $\kappa \in N$, all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{Gen}(1^\kappa)$ and $\rho \xleftarrow{\$} \mathcal{R}$, the distributions of $(\mathsf{sk}', \mathsf{pk}')$ and $(\mathsf{sk}'', \mathsf{pk}'')$ are identical, where:
$$(\mathsf{sk}', \mathsf{pk}') \leftarrow (\mathsf{RandSK}(\mathsf{sk}, \rho), \mathsf{RandPK}(\mathsf{pk}, \rho))$$
and
$$(\mathsf{sk}'', \mathsf{pk}'') \xleftarrow{\$} \mathsf{Gen}(1^\kappa).$$

2. *Correctness under rerandomized keys:* For all $\kappa \in \mathbb{N}$, all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{Gen}(1^\kappa)$, all $\rho \in \mathcal{R}$, and all $m \in \{0,1\}^*$, the rerandomized keys $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$ satisfy:
$$\Pr[\mathsf{Verify}(\mathsf{pk}', \sigma, m) = 1 \mid \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}', m)] = 1.$$

The security notion of *one-per message existential unforgeability under honestly rerandomizable keys* (**uf-cma-hrk1**) differs from the unforgeability notion of standard signature scheme in the following ways: (1) the signing oracle cannot only return signatures under sk, but it can also return signatures that were produced with keys that represent *honest* rerandomizations of sk; (2) the randomness for the rerandomization is chosen uniformly at random from $\mathcal{R}$ by the game; (3) the signing oracle returns at most one signature for each randomness/message pair (ρ, m). The notion of **uf-cma-hrk1** for a rerandomizable signature scheme RSig is formally modeled in the form of a game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$ which we recall in the following definition.

Definition A.1 (One-per message unforgeability under honestly rerandomizable keys of signature schemes with rerandomizable keys). A signature scheme with honestly rerandomizable keys RSig is **uf-cma-hrk1**-secure if no PPT adversary $\mathcal{A}$ wins game **uf-cma-hrk1** as described below with more than advantage. We define $\mathcal{A}$'s advantage in game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$ as $\mathsf{Adv}^{\mathcal{A}}_{\textbf{uf-cma-hrk1}_{\mathsf{RSig}}} := \Pr[\textbf{uf-cma-hrk1}^{\mathcal{A}}_{\mathsf{RSig}} = 1]$.

Game $\textbf{uf-cma-hrk1}_{\mathsf{RSig}}$:

- The challenger initializes two lists as $\mathsf{SigList} \leftarrow \{\epsilon\}$ and $\mathsf{RList} \leftarrow \{\epsilon\}$ and samples a pair of keys $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{RSig.Gen}(1^\kappa)$. Then $\mathcal{A}$ is run on input pk.
- $\mathcal{A}$ is given access to the following oracles:
  - `Rand`: Upon a query, this oracle samples a fresh random value from $\mathcal{R}$ as $\rho \xleftarrow{\$} \mathcal{R}$, stores ρ in RList, and returns ρ.

– `RSign`: On input a message m and a randomness ρ, if ρ was not obtained via a prior `Rand` query (i.e., $\rho \notin \mathsf{RList}$), then this oracle returns $\perp$. Otherwise, it derives a pair of keys rerandomized with the randomness ρ, as $\mathsf{sk}' \leftarrow \mathsf{RSig.RandSK}(\mathsf{sk}, \rho)$ and $\mathsf{pk}' \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}, \rho)$. If $(\mathsf{pk}', m) \in \mathsf{SigList}$ then the oracle returns $\perp$. Otherwise, it derives a signature on message m under the secret key sk' as $\sigma \leftarrow \mathsf{RSig.Sign}(\mathsf{sk}', m)$. The oracle stores the tuple (pk', m) in $\mathsf{SigList}$ and returns σ.

- $\mathcal{A}$ wins if it returns a forgery σ^* together with a message m^* and a public key $\mathsf{pk}^* \leftarrow \mathsf{RSig.RandPK}(\mathsf{pk}, \rho^*)$, s.t. the following holds: (1) the randomness ρ^* has been derived via a `Rand` query, i.e., $\rho^* \in \mathsf{RList}$, (2) $(\mathsf{pk}^*, m^*) \notin \mathsf{SigList}$, and (3) σ^* is a valid forgery, i.e., $\mathsf{RSig.Verify}(\mathsf{pk}^*, \sigma^*, m^*) = 1$.

A.2 TVRF Construction from Galindo et al. [GLOW21]

We briefly recall the TVRF construction from Galindo et al. that is based on the DDH assumption. The construction relies on a non-interactive zero-knowledge proof system (NIZK) for the relation $\mathsf{R} := \{(g, h, X, Y), x \mid X = g^x, Y = h^x\}$ where g and h are two generators of a cyclic group $\mathbb{G}$ of prime order q and $x \in \mathbb{Z}_q$. At a high level, the NIZK proves that two group elements X and Y have the same discrete logarithm w.r.t. generators g and h. This proof system was first introduced by Chaum and Pedersen [CP93] and we denote it by DLEq. We recall the proof system in Figure 4 and the TVRF construction, which we denote by TVRF, in Figure 5. The corruption threshold for the (t, n)-TVRF scheme is set to $t \le \frac{n-1}{2}$.

```
DLEq.Prove($g^x, h^x, x$)
00 Sample $r \xleftarrow{\$} \mathbb{Z}_q$.
01 Compute $c \leftarrow \mathsf{H}(g^x, h^x, g^r, h^r)$
02 Compute $s = r + c \cdot x$.
03 Return $\pi := (c, s)$.

DLEq.Verify($g^x, h^x, \pi$)
00 Parse $\pi := (c, s)$.
01 $R \leftarrow g^s/(g^x)^c$.
02 $R' \leftarrow h^s/(h^x)^c$.
03 If $c \neq \mathsf{H}_1(g^x, h^x, R, R')$: Return 0.
04 Return 1
```

Figure 4: NIZK proof of equality of discrete logarithms with $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$.

```
TVRF.Gen($1^\kappa, t, n$)
00 Sample $a_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [t] \cup \{0\}$
01 Let $F(x) := a_t x^t + \cdots + a_1 x + a_0$
02 $\mathsf{sk} := x \leftarrow a_0 \mod q$, $X \leftarrow g^x$
03 $\mathsf{sk}_i := x_i \leftarrow F(i) \mod q$, $X_i \leftarrow g^{x_i}$
04 $\mathsf{pk} := (X, \{X_1, \cdots, X_n\})$
05 Return $(\mathsf{pk}, \{\mathsf{sk}_1, \cdots, \mathsf{sk}_n\})$

TVRF.Combine($\mathsf{pk}, \mathcal{S}, \{(\phi_i, \pi_i)\}_{i \in \mathcal{S}}$)
00 If $|\mathcal{S}| \le t$: Return $\perp$.
01 Let $\mathcal{S}' := \emptyset$.
02 Parse $\mathsf{pk} := (X, \{X_1, \cdots, X_n\})$.
03 For $i \in \mathcal{S}$, if $\mathsf{DLEq.Verify}(\phi_i, X_i, \pi_i) = 1$:
04     Then $\mathcal{S}' \leftarrow \mathcal{S}' \cup i$.
05 If $|\mathcal{S}'| \le t$: Return $\perp$.
06 $\phi \leftarrow \prod_{i \in \mathcal{S}'} \phi_i^{\lambda_i}$ and $\pi := \{\phi_i, (\pi_i\}_{i \in \mathcal{S}'}$.
07 Return $(\phi, \pi)$.

TVRF.PEval($m, \mathsf{sk}_i, \mathsf{pk}$)
00 Parse $\mathsf{pk} := (X, \{X_1, \cdots, X_n\})$
01 $\phi_i \leftarrow \mathsf{H}_1(m)^{\mathsf{sk}_i}$.
02 $\pi_i \leftarrow \mathsf{DLEq.Prove}(\phi_i, X_i, \mathsf{sk}_i)$.
03 Return $(\phi_i, \pi_i)$.

TVRF.Verify($\mathsf{pk}, m, \phi, \pi$)
00 Parse $\mathsf{pk} := (X, \{X_1, \cdots, X_n\})$.
01 Parse $\pi := \{\phi_i, (\pi_i\}_{i \in \mathcal{S}'}$.
02 Let $\mathcal{S}' := \emptyset$.
03 For $i \in \mathcal{S}'$:
04     if $\mathsf{DLEq.Verify}(\phi_i, X_i, \pi_i) \neq 1$
05         return $\perp$.
06 If $\phi = \prod_{i \in \mathcal{S}'} \phi_i^{\lambda_i}$: Return 1.
07 Else return 0.
```

Figure 5: Threshold verifiable random function from [GLOW21] for a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q and for a cryptographic hash function $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}$.

A.3 ECDSA with Rerandomizable Keys

We briefly recall the standard ECDSA signature scheme in Figure 6 and then describe how it can be extended to achieve the ECDSA-based signature scheme with additively rerandomizable keys as shown in [DEF+21].

The ECDSA signature scheme is defined for a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q where the discrete logarithm problem in $\mathbb{G}$ is hard. We briefly recall the scheme here, which we denote by $\mathsf{ECDSA}[\mathsf{H}]$, where $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ is a cryptographic hash function.

```
$\mathsf{Gen}(1^\kappa)$
$x \xleftarrow{\$} \mathbb{Z}_q$
$X \leftarrow g^x$
$(\mathsf{sk}, \mathsf{pk}) := (x, X)$
Return $(\mathsf{sk}, \mathsf{pk})$
```

```
$\mathsf{Sign}(\mathsf{sk}, m)$
Parse $\mathsf{sk} := x$
$k \xleftarrow{\$} \mathbb{Z}_q, R \leftarrow g^k$
If $R = 1$: Return $\perp$
$r \leftarrow f(R)$
If $r = 0$: Return $\perp$
$h \leftarrow \mathsf{H}(m)$
$s = k^{-1}(h + r \cdot x)$
If $s = 0$: Return $\perp$
Return $\sigma := (r, s)$
```

```
$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
Parse $\mathsf{pk} := X$ and $\sigma := (r, s)$
If $s = 0 \lor t = 0$: Return $\perp$
$h \leftarrow \mathsf{H}_0(m)$
$u_1 \leftarrow h \cdot s^{-1}$
$u_2 \leftarrow r \cdot s^{-1}$
$R \leftarrow g^{u_1} + X^{u_2}$
If $f(R) = r$: Return 1
Return 0
```

Figure 6: ECDSA signature scheme $\mathsf{ECDSA}[\mathsf{H}]$ instantiated with a cryptographic hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$.

In Figure 7, we recall the ECDSA-based signature scheme with rerandomizable keys $\mathsf{rECDSA}[\mathsf{H}]$ as introduced in [DEF+21].

B The GG scheme by Gennaro and Goldfeder [GG18]

B.1 Underlying Assumptions and Building Blocks

Decisional Diffie-Hellman Problem (DDH) Let $\mathbb{G}$ be a cyclic group of prime order q and let g be a generator of $\mathbb{G}$. Let a, b, c be elements chosen uniformly at random from $\mathbb{Z}_q$. Then the distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable.

Non-interactive zero knowledge proof (NIZK) A NIZK proof of knowledge with respect to a polynomial-time recognizable binary relation R is given by the following tuple of PPT algorithms $\mathsf{ZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, where (i) $\mathsf{Setup}(1^\kappa)$ outputs a common reference string crs; (ii) $\mathsf{Prove}(\mathsf{crs}, (Y, y))$ outputs a proof π for $(Y, y) \in \mathsf{R}$; (iii) $\mathsf{Verify}(\mathsf{crs}, Y, \pi)$ outputs a bit $b \in \{0,1\}$. Further, the NIZK proof of knowledge w.r.t. R should satisfy the following properties:

1. *Completeness*: For all $(Y, y) \in \mathsf{R}$, all $\kappa \in \mathbb{N}$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$, it holds that $\mathsf{Verify}(\mathsf{crs}, Y, \mathsf{Prove}(\mathsf{crs}, (Y, y))) = 1$ except with negligible probability;

2. *Soundness*: For any $(Y, y) \notin \mathsf{R}$, all $\kappa \in \mathbb{N}$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$, it holds that $\mathsf{Verify}(\mathsf{crs}, Y, \mathsf{Prove}(\mathsf{crs}, (Y, y))) = 0$ except with negligible probability;

3. *Zero knowledge*: For any PPT adversary $\mathcal{A}$, there exist a PPT algorithm $\pi_{\mathcal{S}} \leftarrow \mathcal{S}(\mathsf{crs}, Y)$ such that for all $\kappa \in \mathbb{N}$, all $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$ and all $(Y, y) \in \mathsf{R}$, the distributions $\{(\pi, Y) : \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, Y, y)\}$ and $\{(\pi_{\mathcal{S}}, Y) : \pi_{\mathcal{S}} \leftarrow \mathcal{S}(\mathsf{crs}, Y)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability.

Non-Malleable and Equivocable Commitments A non-malleable and equivocable commitment scheme with message space $\{0,1\}^*$, commitment space $\mathcal{C}$ and opening space $\mathcal{O}$ consists of a tuple of three PPT algorithms $\mathsf{CT} := (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Equivocate})$ where Gen gets as input the security parameter $\kappa \in \mathbb{N}$ and outputs public parameters par and a trapdoor τ; Com takes as input par and a message $m \in \{0,1\}^*$ and outputs a tuple (c, d); Open takes as input par and a tuple $(c, d) \in (\mathcal{C} \times \mathcal{O})$ and either

```
Sign (sk, $m$)
00 $m' \leftarrow (\mathsf{pk}, m)$
01 $\sigma \leftarrow \mathsf{ECDSA[H].Sign}(\mathsf{sk}, m')$
02 Return $\sigma$

Verify (pk, $\sigma$, $m$)
03 $m' \leftarrow (\mathsf{pk}, m)$
04 Return
$\mathsf{ECDSA[H].Verify}(\mathsf{pk}, m', \sigma)$

RandSK (sk, $\rho$)
00 $\mathsf{sk}' \leftarrow (\mathsf{sk} + \rho) \mod q$
01 Return $\mathsf{sk}'$

RandPK (pk, $\rho$)
02 $\mathsf{pk}' \leftarrow (\mathsf{pk} + g^{\rho})$
03 Return $\mathsf{pk}'$
```

Figure 7: Public key prefixed version of the ECDSA signature scheme with perfectly rerandomizable keys $\mathsf{rECDSA[H]}$ based on the ECDSA signature scheme $\mathsf{ECDSA[H]}$. Above $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_q$ denotes a cryptographic hash function.

outputs a message m or $\perp$; $\mathsf{Equivocate}$ takes as input a trapdoor τ, a commitment $c \in \mathcal{C}$ and a message $m \in \{0,1\}^*$ and outputs an opening d. A non-malleable and equivocable commitment scheme must satisfy the following properties:

1. *Computationally Hiding*: For all $\kappa \in \mathbb{N}$, all $(\mathsf{par}, \tau) \leftarrow \mathsf{Gen}(1^\kappa)$, any two messages $m, m' \in \{0,1\}^*$ and $(c, d) \leftarrow \mathsf{Com}(\mathsf{par}, m)$ and $(c', d') \leftarrow \mathsf{Com}(\mathsf{par}, m')$, there exists no PPT adversary $\mathcal{A}$ which can distinguish the tuples (m, m', c) and (m, m', c') except with negligible probability.

2. *Computationally Binding*: For all $\kappa \in \mathbb{N}$ and all $(\mathsf{par}, \tau) \leftarrow \mathsf{Gen}(1^\kappa)$, there exists no PPT adversary $\mathcal{A}$ which can output (c, d, d') such that $\mathsf{Open}(\mathsf{par}, c, d) \neq \mathsf{Open}(\mathsf{par}, c, d')$ and $\mathsf{Open}(\mathsf{par}, c, d) \neq \perp$ and $\mathsf{Open}(\mathsf{par}, c, d') \neq \perp$ except with negligible probability.

3. *Equivocable*: For all $\kappa \in \mathbb{N}$, all $(\mathsf{par}, \tau) \leftarrow \mathsf{Gen}(1^\kappa)$ and any message $m \in \{0,1\}^*$ the distributions $\{(c, d) : (c, d) \leftarrow \mathsf{Com}(\mathsf{par}, m)\}$ and $\{(c', d') : c' \xleftarrow{\$} \mathcal{C}, d' \leftarrow \mathsf{Equivocate}(\tau, c', m)\}$ are computationally indistinguishable.

Finally, a commitment scheme is *non-malleable* if for all $\kappa \in \mathbb{N}$, all $(\mathsf{par}, \tau) \leftarrow \mathsf{Gen}(1^\kappa)$, any message $m \in \{0,1\}^*$ and $(c, d) \leftarrow \mathsf{Com}(\mathsf{par}, m)$, there exists no PPT adversary $\mathcal{A}$ which on input c can output a commitment c' such that after receiving the opening d the adversary $\mathcal{A}$ can output an opening d' such that for $m' \leftarrow \mathsf{Open}(\mathsf{par}, c', d')$ the messages m and m' are related.

Public Key Encryption A public key encryption scheme consists of three algorithms $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where (i) $\mathsf{Gen}(1^\kappa)$ outputs a public key pk and a secret key sk; (ii) $\mathsf{Enc}(\mathsf{pk}, m)$ outputs a ciphertext ct; and (iii) $\mathsf{Dec}(\mathsf{sk}, ct)$ outputs either $\perp$ or a message m.

A public key encryption scheme $\mathsf{pk} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is linearly homomorphic if (1) there exists an efficiently computable operation $\oplus$ s.t. for two ciphertexts $ct_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$ and $ct_2 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_2)$ it holds that $ct_1 \oplus ct_2 = \mathsf{Enc}(\mathsf{pk}, m_1 + m_2)$; and (2) there exists an efficiently computable operation $\odot$ s.t. for a ciphertext $ct_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$ and a constant k it holds that $ct_1 \oplus k = \mathsf{Enc}(\mathsf{pk}, m_1 \cdot k)$.

A public key encryption scheme is semantically secure if for every PPT adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function ν in the security parameter $\kappa \in \mathbb{N}$ s.t.:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \\ (m_1, m_2, s) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ b \xleftarrow{\$} \{0,1\}, ct \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b), \\ b' \leftarrow \mathcal{A}_2(s, ct) \end{array} \right] \leq 1/2 + \nu(\kappa).$$

B.2 Construction

The $\mathsf{GG}[\mathsf{H}_0]$ scheme relies on a multiplicative to additive share conversion protocol, which allows two parties P_i and P_j with shares $a_i \in \mathbb{Z}_q$ and $b_j \in \mathbb{Z}_q$ respectively s.t. $x = a_i \cdot b_j \mod q$ to transform a_i and b_j into additive shares of x, i.e., into shares α_i and β_j s.t. $x = \alpha_i + \beta_j$. We briefly recall this protocol here. We denote by PKE a linearly homomorphic encryption scheme (with operations $\odot$ for multiplication with a constant and $\oplus$ for homomorphic addition) over an integer N and we denote by $(\mathsf{pk}_{i,\mathsf{PKE}}, \mathsf{sk}_{i,\mathsf{PKE}})$ the public/secret key pair of scheme PKE of P_i.

$P_i(\mathsf{pk}_{j,\mathsf{PKE}}, \mathsf{sk}_{i,\mathsf{PKE}}, a_i)$		$P_j(\mathsf{pk}_{i,\mathsf{PKE}}, b_j)$
$c_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,\mathsf{PKE}}, a_i)$ Compute a ZK proof π_i that $a < q^3$	$\xrightarrow{c_i, \pi_i}$	
		If π_i is not valid, abort. $y \xleftarrow{\$} \mathbb{Z}_{q^5}$, $c_j \leftarrow (c_i \odot b_j) \oplus \mathsf{PKE.Enc}(\mathsf{pk}_{i,\mathsf{PKE}}, y)$ Compute a ZK proof π_j that P_j knows $b_j < q^3$, $y < q^7$ s.t. $c_j \leftarrow (c_i \odot b_i) \oplus \mathsf{PKE.Enc}(\mathsf{pk}_{i,\mathsf{PKE}}, y)$
	$\xleftarrow{c_j, \pi_j}$	
If π_j is not valid, abort. $\alpha \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{i,\mathsf{PKE}}, c_j) \mod q$		$\beta \leftarrow -y \mod q$

Figure 8: Multiplicative to additive share conversion protocol MtA.

Gennaro and Goldfeder also consider a slight adjustment of the above protocol which they call MtAwc, which differs only from the above protocol in the following way: If $B_j = g^{b_j}$ is a public value (where g is the generator of a cyclic group of prime order q), then party P_j additionally proves in zero-knowledge that b_j is the discrete log of B_j. We now recall the key generation and signing procedures of the $\mathsf{GG}[\mathsf{H}_0]$ scheme. For simplicity, we slightly deviate from the original $\mathsf{GG}[\mathsf{H}_0]$ scheme in two ways. We emphasize that these two changes have no impact on the scheme's security: Gennaro and Goldfeder consider a distributed key generation, whereas we assume that the key generation is initially executed by a trusted party. In addition, we do not generate the keys for the linearly homomorphic encryption scheme during the initial key generation but we let parties generate fresh keys in the beginning of an execution of the signing procedure.

```
Algorithm $\mathsf{Gen}(1^\kappa, t, n)$
00 For $k \in [t] \cup \{0\}$, sample $a_k \xleftarrow{\$} \mathbb{Z}_q$.
01 Let $F(x) := a_t x^t + \cdots + a_1 x + a_0$.
02 $\mathsf{sk} := x \leftarrow a_0 \mod q$.
03 Set $X \leftarrow g^x$ and $\mathsf{sk}_i := x_i \leftarrow F(i) \mod q$.
04 Set $X_i \leftarrow g^{x_i}$.
05 Set $\mathsf{pk} := (X, \{X_i\}_{i \in [n]})$.
06 Return $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in [n]})$.
```

Figure 9: Key generation algorithm. Note that Gennaro and Goldfeder consider a distributed key generation, whereas we assume that the key generation is initially executed by a trusted party.

In Figure 10 we recall the signing procedure of the $\mathsf{GG}[\mathsf{H}_0]$ scheme. The procedure makes use of a non-malleable and equivocable commitment scheme $\mathsf{CT} := (\mathsf{Com}, \mathsf{Open})$ as well as a hash function $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_q$, a linearly homomorphic encryption scheme $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-interactive zero-knowledge proof system ZK. We slightly adjust the signing procedure as follows: Instead of letting parties generate their key pair for PKE during the initial key generation, we let parties generate a fresh key pair $(\mathsf{pk}_{i,\mathsf{PKE}}, \mathsf{sk}_{i,\mathsf{PKE}})$ for the PKE scheme before Phase 1 of the signing procedure. Each party then broadcasts $\mathsf{pk}_{i,\mathsf{PKE}}$ together with a zero-knowledge proof that the key was generated honestly.[9] The parties then engage in the signing procedure as specified in Figure 10.

[9]We note that if PKE is instantiated with the Paillier encryption scheme [Pai99], then this zero-knowledge proof can be instantiated with the Paillier-Blum Modulus zero-knowledge proof system [CGG+20b].

$P_i(w_i, m)$ **Phase 1** $P_j(w_j, m)\{j \neq i\}$

$k_i \xleftarrow{\$} \mathbb{Z}_q,\ \gamma_i \xleftarrow{\$} \mathbb{Z}_q$
$(C_i, D_i) \leftarrow \mathsf{CT.Com}(g^{\gamma_i})$

$\xrightarrow{C_i}$

Define the following:
$k = \sum_{i \in S} k_i,\ \gamma = \sum_{i \in S} \gamma_i$
$k\gamma = \sum_{i,j \in S} k_i \gamma_j \mod q$
$kx = \sum_{i,j \in S} k_i w_j \mod q$

Phase 2

$$\begin{tikzcd}
{} \arrow[r, "k_i"] & \boxed{\mathsf{MtA}} \arrow[l, "\alpha_{i,j}", shift left=2] \arrow[r, "\beta_{i,j}", shift right=2] & {} \arrow[l, "\gamma_j"'] \\
{} \arrow[r, "k_i"] & \boxed{\mathsf{MtAwc}} \arrow[l, "\mu_{i,j}", shift left=2] \arrow[r, "\nu_{i,j}", shift right=2] & {} \arrow[l, "w_j"']
\end{tikzcd}$$

(s.t. $k_i \cdot \gamma_j = \alpha_{i,j} + \beta_{j,i}$)
(s.t. $k_i \cdot w_j = \mu_{i,j} + \nu_{j,i}$)

$\delta_i = k_i\gamma_i + \sum_{j \neq i}(\alpha_{i,j} + \beta_{j,i})$
$\sigma_i = k_i w_i + \sum_{j \neq i}(\mu_{i,j} + \nu_{j,i})$

Phase 3

$\xrightarrow{\delta_i}$ $\delta = \sum_{i \in S} \delta_i = k\gamma$

Phase 4

$\pi_{\gamma_i} = \mathsf{ZK}_{\Gamma_i}\{(\gamma_i) \colon \Gamma_i = g^{\gamma_i}\}$ $\xrightarrow{D_i, \pi_{\gamma_i}}$ $\Gamma_i = \mathsf{CT.Open}(C_i, D_i)$
Abort if π_{γ_i} does not verify
$R = \left(\prod_{i \in S} \Gamma_i\right)^{\delta^{-1}} = g^{k^{-1}}$,
where $R = (r_x, r_y)$.
Set $r = r_x \mod q$

Phase 5

$m' = \mathsf{H}_0(m),\ s_i = m'k_i + r\sigma_i$
$l_i \xleftarrow{\$} \mathbb{Z}_q,\ \rho_i \xleftarrow{\$} \mathbb{Z}_q$
$V_i = R^{s_i} \cdot g^{l_i},\ A_i = g^{\rho_i}$

$(\hat{C}_i, \hat{D}_i) = \mathsf{CT.Com}(V_i, A_i)$ $\xrightarrow{\hat{C}_i}$

$\hat{\pi}_i = \mathsf{ZK}_{(V_i, A_i)}\{(s_i, l_i, \rho_i) \colon$ $\xrightarrow{\hat{D}_i, \hat{\pi}_i}$
$(V_i = R^{s_i} \cdot g^{l_i}) \wedge (A_i = g^{\rho_i})\}$

Abort if a proof fails
$V = g^{-m'} \cdot Q^{-r} \cdot \prod_{i \in S} V_i = g^l$
$A = \prod_{i \in S} A_i$

$U_i = V^{\rho_i},\ T_i = A^{l_i}$

$(\tilde{C}_i, \tilde{D}_i) = \mathsf{CT.Com}(U_i, T_i)$ $\xrightarrow{\tilde{C}_i}$

$\xrightarrow{\tilde{D}_i}$ Abort if $\sum_{i \in S} T_i \neq \sum_{i \in S} U_i$

$\xrightarrow{s_i}$ $s = \sum_{i \in S} s_i$

Figure 10: Interactive (t, n)-threshold ECDSA scheme by Gennaro and Goldfeder [GG18], where $|S| \subseteq [n], |S| = t + 1$. For all parties $\{P_i\}_{\{i \in [n]\}}$, x_i denotes secret share of the secret x. For all parties $P_{i\{i \in S\}}$, w_i represents the secret share of x due to $(t, t + 1)$-secret sharing of x, such that $x = \sum_{i \in S} w_i$.

Finally, in Figure 11 we recall the simulation of the signing procedure as provided in [GG18] (with some minor modifications). The forger $\mathcal{F}$ provides a computationally indistinguishable view of the signing procedure of the GG scheme to a PPT adversary on input the secret key shares of corrupted parties and with access to a signing oracle.

Simulation of the Signing Procedure: Before Phase 1 of the signing procedure, $\mathcal{F}$ samples uniformly at random a public key $\mathsf{pk}_{1,\mathsf{PKE}}$ s.t. $(\mathsf{pk}_{1,\mathsf{PKE}}, \cdot) \in \mathsf{PKE.Gen}(1^\kappa)$ and simulates the zero-knowledge proof $\pi_{1,\mathsf{PKE}}$.

Phase 1: $\mathcal{F}$ executes Phase 1 honestly for party P_1, i.e., it samples $k_1, \gamma_1 \xleftarrow{\$} \mathbb{Z}_q$ and commits to g^{γ_1}. It then broadcasts the commitment C_1.

Phase 2: $\mathcal{F}$ executes the first MtA protocol correctly for P_1 using the values k_1 and γ_1 and extracts the following values from the zero-knowledge proofs that are exchanged during the MtA protocol: k_i, γ_i, y_1 for $i > 1$. It then computes $\alpha_{1,j} = k_1\gamma_i + y_1 \mod q$ and $\tilde{k} = \sum_{i \in S} k_i \mod q$.

For the execution of the MtAwc protocol, $\mathcal{F}$ does not know w_1 when P_1 is the reacting party. Therefore, it simply chooses a random $\gamma_{j,1}$ and simulates the corresponding zero-knowledge proofs. When P_1 is the initiating party, $\mathcal{F}$ can execute the protocol honestly with input k_1 and extract the share $\nu_{1,j}$ from the zero-knowledge proofs.

Phase 3: $\mathcal{F}$ executes this phase correctly for P_1.

Phase 4: All players decommit to Γ_i. $\mathcal{F}$ extracts γ_j for all j_1 from the zero-knowledge proofs π_{γ_j} and computes $k = \delta \cdot \left(\sum_{i>0} \gamma_i\right)^{-1} \mod q$.

If $\tilde{k} = k$, then $\mathcal{F}$ proceeds as follows:

(a) $\mathcal{F}$ queries its own signing oracle on message m to receive a signature (r, s) and computes $R = g^{\mathsf{H}(m)s^{-1}} \cdot X^{rs^{-1}}$.

(b) $\mathcal{F}$ rewinds the adversary to the beginning of Phase 4 and equivocates the decommitment of P_1 to $\hat{\Gamma_1} = R^\delta \prod_{i>1} \Gamma_i^{-1}$.

(c) $\mathcal{F}$ computes $s_1 = s - \sum_{i>1} s_i$.

Phase 5: $\mathcal{F}$ executes this phase correctly for P_1 using s_1.

Else if $\tilde{k} \neq k$, then $\mathcal{F}$ proceeds as follows:

Phase 4: $\mathcal{F}$ runs this phase correctly for P_1.

Phase 5: $\mathcal{F}$ chooses $\tilde{s_1} \xleftarrow{\$} \mathbb{Z}_q$ and runs this phase using this value.

Figure 11: Simulation of the signing procedure of the GG scheme. The forger $\mathcal{F}$ receives as input the secret key shares of all corrupted parties and obtains access to a signing oracle.

B.3 Proof Sketch of Theorem 4.2

B.3.1 Case $b = 0$

In this case, $\mathcal{B}$ executes $\mathcal{B}_0$ which plays in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$. That is, upon $\mathcal{A}$ sending the list C of parties to corrupt in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$, $\mathcal{B}_0$ corrupts the same parties in $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ and forwards the resulting secret key shares and the public key to $\mathcal{A}$.

The simulation of the oracles `Rand`, `RSign` and the random oracle H_0 happens in a straightforward way, i.e., $\mathcal{B}_0$ simply forwards queries from $\mathcal{A}$ in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ to the corresponding oracle in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$.

The simulation of the random oracle H_1 and the `REval` oracle is a bit more challenging as $\mathcal{B}_0$ does not have access to any such oracle in game $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$. Upon a query from $\mathcal{A}$ to the random oracle H_1 on input a message m, $\mathcal{B}_0$ first checks if $\mathsf{H}_1(m)$ has been set already. If so, it simply returns $\mathsf{H}_1(m)$. Otherwise it samples a uniformly random value $r \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathsf{H}_1(m) := g^r$ and returns g^r. The simulation of the `REval` oracle then works as follows: On input a message m, an index $i \in [n]$ and a randomness $\rho \in \mathsf{RList}$, $\mathcal{B}_0$ first executes $\mathsf{pk}' \leftarrow \mathsf{TVRF\text{-}rGG.RandPK}(\mathsf{pk}, \rho)$ and parses $\mathsf{pk}' := (X', \{X'_1, \cdots, X'_n\})$. $\mathcal{B}_0$ then retrieves $r \leftarrow \mathsf{H}_1(m)$, sets $\phi_i := (X'_i)^r = \mathsf{H}_1(m)^{\mathsf{sk}'_i}$, simulates the corresponding zero-knowledge proof π_i and returns (ϕ_i, π_i).

Eventually, the adversary outputs a forgery which $\mathcal{B}_0$ also forwards to the $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ game. It is easy to see that $\mathcal{B}_0$ wins the $\textbf{th-ufcma-hrk1}_{\mathsf{rGG}}$ game if $\mathcal{A}$ is able to win the $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$ game by satisfying the winning condition in **Case 1**.

B.3.2 Case $b = 1$

In this case, $\mathcal{B}$ executes $\mathcal{B}_1$ which plays in game $\textbf{th-prand}_{\mathsf{TVRF}}$. That is, upon $\mathcal{A}$ sending the list C of parties to corrupt in game $\textbf{unf-prand}_{\mathsf{TVRF\text{-}rGG}}$, $\mathcal{B}_1$ corrupts the same parties in $\textbf{th-prand}_{\mathsf{TVRF}}$ and forwards the resulting secret key shares and the public key to $\mathcal{A}$. The simulation of oracles `Rand`, `RSign`, H_0, H_1 and `REval` then works as follows:

- **Oracle `Rand`:** On a query to `Rand` from $\mathcal{A}$, $\mathcal{B}_1$ samples uniformly at random $\rho \xleftarrow{\$} \mathbb{Z}_q$, stores ρ in RList and returns ρ.
- **Oracle H_0:** Upon $\mathcal{A}$ querying H_0 on input a message m, $\mathcal{B}_1$ first checks whether m is public key prefixed, i.e., whether m can be parsed as $m := (\mathsf{pk}', m')$ where $(\mathsf{pk}', \cdot) \in \mathsf{TVRF\text{-}rGG.Gen}(1^\kappa, t, n)$. If so, $\mathcal{B}_1$ executes $\mathcal{S}_{\mathsf{ECDSA}}$ as described in Figure 3 on input (X', m) with $\mathsf{pk}' := (X', \{X'_1, \cdots, X'_n\})$.

  If m is not public key prefixed, $\mathcal{B}_1$ simply samples a uniformly random value $r \xleftarrow{\$} \mathbb{Z}_q$, sets $\mathsf{H}_0(m) := r$ and returns $\mathsf{H}_0(m)$. Note that in order for $\mathcal{B}_1$ to abort in this simulation, $\mathcal{A}$ would have to guess a randomness $\rho \in \mathbb{Z}_q$ before it has been output by the `Rand` oracle. This happens only with negligible probability. Further, note that $\mathcal{S}_{\mathsf{ECDSA}}$ programs the random oracle H_0 in such a way that (1) $\mathsf{H}_0(m)$ is set to uniform random value in $\mathbb{Z}_q$, and (2) the values (r, s) look like a valid ECDSA signature for m and X' to $\mathcal{A}$ except with negligible probability (this has been shown in [FKP17]).
- **Oracle `RSign`:** Upon $\mathcal{A}$ querying this oracle on input a message m and randomness $\rho \in \mathsf{RList}$, $\mathcal{B}_1$ simulates the signing procedure in the same way as described in Theorem 3.4. Note that this simulation relies on the availability of a signing oracle, which returns full valid ECDSA signatures on arbitrary messages and rerandomized public keys. Since $\mathcal{B}_1$ does not have access to such an oracle, it uses the simulated signatures (r, s) that are generated during the programming of H_0. Note that the simulator code from Theorem 3.4 does not program H_0 such that there is no conflict between the execution of the simulator code from Theorem 3.4 and $\mathcal{S}_{\mathsf{ECDSA}}$.
- **Oracle H_1:** Upon $\mathcal{A}$ querying H_1 on some message m, $\mathcal{B}_1$ simply queries its own random oracle on m and relays the output.
- **Oracle `REval`:** Upon a query from $\mathcal{A}$ on input (m, i, ρ), $\mathcal{B}_1$ queries its own oracle on input m and receives an evaluation share (ϕ_i, π_i) where $\phi_i = \mathsf{H}_1(m)^{\mathsf{sk}_i}$. $\mathcal{B}_1$ then computes $\phi'_i = \phi_i \cdot H_1(m)^{\rho_i} = H_1(m)^{\mathsf{sk}_i + \rho_i}$ (where ρ_i is the randomness share of ρ for party P_i according to the $\mathsf{TVRF\text{-}rGG.RandSK}$ algorithm), simulates a NIZK proof π'_i of the DLEq proof system (cf. Appendix A.2) and sends (ϕ'_i, π'_i) to $\mathcal{A}$.

<u>Reduction to $\textbf{th-prand}_{\mathsf{TVRF}}$</u>: During the challenge phase of $\textbf{th-prand}_{\mathsf{TVRF\text{-}rGG}}$ (in **Case 2**), $\mathcal{A}$ outputs a message m^*, randomness ρ^*, a set of indices $\mathcal{S}$ and evaluation shares $\{(\phi^*_i, \pi^*_i)\}_{i \in \mathcal{S} \cap \mathsf{C}}$. Upon receiving these values, the adversary $\mathcal{B}_1$ computes $\phi_i = \phi^*_i \cdot \mathsf{H}_1(m^*)^{-\rho^*_i} = \mathsf{H}_1(m^*)^{\mathsf{sk}_i}$ and generates a new zero-knowledge proof π_i using sk_i and ϕ_i. $\mathcal{B}_1$ then returns the set of indices $\mathcal{S}$, the message m^* and evaluation shares $\{(\phi_i, \pi_i\}_{i \in \mathcal{S} \cap \mathsf{C}}$ to game $\textbf{th-prand}_{\mathsf{TVRF}}$. Upon receiving the challenge value ϕ from the underlying game, $\mathcal{B}_1$ computes $\phi^* = \phi \cdot \mathsf{H}_1(m^*)^{\rho^*}$ and returns it to $\mathcal{A}$. Note that if ϕ was chosen randomly by the $\textbf{th-prand}_{\mathsf{TVRF}}$ game then ϕ^* is also random, and if ϕ is a valid TVRF output, then so is ϕ^* under the key randomized with ρ^*. $\mathcal{B}_1$ then simply relays the output of $\mathcal{A}$ to its own game.

It is easy to see that if $\mathcal{A}$ can distinguish between a random value and the output of the rerandomized TVRF, $\mathcal{B}_1$ can distinguish between a random value and the output of the TVRF.

C. Deterministic Wallets in a Quantum World

In this chapter, we present an adjusted version of the following publication:

[5] N. A. Alkadri, P. Das, A. Erwig, S. Faust, J. Krämer, S. Riahi, and P. Struck. "Deterministic Wallets in a Quantum World". In: *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.* 2020, pp. 1017–1031. **Part of this thesis.**

Concretely, the work included in this chapter corrects the following mistake that is contained in the above publication [5]: In [5] the rerandomizability of public keys property of a signature scheme with rerandomizable public keys required a rerandomized public key to be identically distributed to a freshly generated public key. However, in our construction, the distribution of a rerandomized public key is only computationally indistinguishable from a freshly generated public key. The work included in this chapter fixes this mistake by adjusting the definition accordingly and by making all subsequently required changes throughout the paper. In particular, this includes the definition of the simulatability property of a signature scheme with rerandomizable public keys, which is required for the wallet unlinkability proof of our generic wallet construction.

Deterministic Wallets in a Quantum World

Nabil Alkeilani Alkadri[1], Poulami Das[2], Andreas Erwig[2], Sebastian Faust[2], Juliane Krämer[3], Siavash Riahi[2], and Patrick Struck[3]

[1] CDC, Technische Universität Darmstadt, Germany
`nabil.alkadri@tu-darmstadt.de`
[2] CAC, Technische Universität Darmstadt, Germany
`{poulami.das,andreas.erwig,sebastian.faust,siavash.riahi}@tu-darmstadt.de`
[3] QPC, Technische Universität Darmstadt, Germany
`{juliane,patrick}@qpc.tu-darmstadt.de`

Abstract. Most blockchain solutions are susceptible to quantum attackers as they rely on cryptography that is known to be insecure in the presence of quantum adversaries. In this work we advance the study of quantum-resistant blockchain solutions by giving a quantum-resistant construction of a deterministic wallet scheme. Deterministic wallets are frequently used in practice in order to secure funds by storing the sensitive secret key on a so-called *cold wallet* that is not connected to the Internet. Recently, Das et al. (CCS'19) developed a formal model for the security analysis of deterministic wallets and proposed a generic construction from certain types of signature schemes that exhibit key rerandomization properties. We revisit the proposed classical construction in the presence of quantum adversaries and obtain the following results.

First, we give a generic wallet construction with security in the quantum random oracle model (QROM) if the underlying signature scheme is secure in the QROM. We next design the first post-quantum secure signature scheme with rerandomizable public keys by giving a construction from generic lattice-based Fiat-Shamir signature schemes. Finally, we show and evaluate the practicality by analyzing an instantiation of the wallet scheme based on the signature scheme qTESLA (ACNS'20).

Keywords: blockchain protocols · deterministic wallets · post-quantum · rerandomizable signatures · provable security · lattice-based cryptography

1 Introduction

In the past decade cryptocurrencies such as Ethereum [21] and Bitcoin [36] have gained huge popularity introducing a revolutionary payment paradigm. Cryptocurrencies do not rely on any central authority (i.e., banks) for the validation of transactions but instead use a consensus protocol to reach agreement on the validity of transactions in a decentralized network. As the name suggests the security of cryptocurrencies heavily relies on cryptographic building blocks – most importantly, on digital signature schemes. Digital signatures are used to authenticate money transfers between parties, where each party is identified by a public key with respect to the signature scheme. In a nutshell, a transfer of v coins from sender pk_{S} to receiver pk_{R} is represented by a transaction $\mathsf{tx} := (pk_{\mathrm{S}}, pk_{\mathrm{R}}, v)$. The transaction tx is then sent together with a signature of tx with respect to the sender's public key pk_{S} to the network of miners who validate the transaction. Besides digital signatures many other (partially advanced) cryptographic building blocks are used by cryptocurrencies to achieve a variety of goals. This includes, for instance, non-interactive zero-knowledge proofs and ring signatures for privacy preserving transactions [20, 38], threshold signatures and deterministic wallets for securing funds [16], aggregate signatures for scalability [26], and many more [22, 25, 42].

Unfortunately, most cryptographic primitives used by cryptocurrencies today can be broken by quantum adversaries. Most notably, the ECDSA signature scheme that is implemented by nearly all popular cryptocurrencies relies on the hardness of computing discrete logarithms, and hence can be broken by Shor's algorithm [40]. Since quantum computers can have devastating consequences for the security of cryptocurrencies [6], several recent works design cryptocurrencies with post-quantum security features, i.e., they resist both classical and quantum attacks, but run on classical machines. Cryptocurrency projects such as "Bitcoin Post Quantum" [1] or QRL [2] replace ECDSA with hash-based post-quantum signatures. Other examples include a Monero-based cryptocurrency with privacy guarantees that hold against quantum adversaries [20], or a security analysis of the proof of work consensus protocol in the quantum random oracle model [15]. In this work, we follow this line of work and investigate the post-quantum security of deterministic wallet schemes, and propose the first construction that provably resists quantum adversaries.

Deterministic wallets. In cryptocurrencies, secret keys are a particular attractive target for attackers. Indeed, the most devastating attacks in the cryptocurrency space have typically targeted secret keys of users resulting in billions of dollars worth of cryptocurrency being stolen [11, 12, 41]. To protect keys against theft, one of the most prominent solutions is the concept of a *deterministic wallet*. A deterministic wallet scheme consists of two components: a hot wallet that is permanently connected to the Internet, and a cold wallet, which comes online only rarely (e.g., when a large amount of money has to be transferred). Das et al. [16] formalized the concept of deterministic wallets and defined its security goals. The first security goal is *wallet unforgeability* which states that funds sent to the cold wallet must remain secure even if the hot wallet is corrupted. Second, *wallet unlinkability*, which guarantees that individual transactions that sent money to the same wallet are unlinkable despite being publicly available on the blockchain.

At a high-level a hot/cold wallet scheme works as follows. In an initialization phase, it generates a master key pair (msk, mpk), where the master secret key msk is stored on the cold wallet, while the hot wallet keeps the corresponding master public key mpk. The main ingredient of a deterministic wallet scheme is a deterministic key derivation procedure, which allows both, cold and hot wallet, to derive matching secret and public session keys without interacting with each other. To this end, in addition to the master secret/public key, the hot and cold wallet share a state St. From this state each wallet can derive the corresponding session key by combining the master key with a deterministically derived value $\mathsf{H}(St, ID)$, where ID is an arbitrary key identifier and H is a cryptographic hash function. More concretely, consider a simplified version of the BIP32 deterministic wallet scheme [3] used for Bitcoin. The master secret/public key pair consists of a valid ECDSA key pair $(msk, mpk) := (x, x \cdot G)$, where G is a generator of the ECDSA elliptic curve. The session key pair for identity ID is computed as $pk_{ID} := mpk + w \cdot G$ and $sk := msk + w$ with $w := \mathsf{H}(St, ID)$.

Post-quantum security of deterministic wallets. While deterministic wallet schemes offer an elegant solution to increase the security of users' funds, they are particularly susceptible to attacks by quantum adversaries. To illustrate this, let us first consider how quantum attacks against the underlying signature scheme of a cryptocurrency such as Bitcoin work. Recall that in Bitcoin (as in most other cryptocurrencies) an address for transferring funds to is not represented by the public key itself but by its hash value. More concretely, when a party transfers v coins to some receiver R with public key pk_{R}, then the transaction will store $h = \mathsf{H}(pk_{\mathrm{R}})$. Only when R wants to spend these coins he reveals pk_{R} together with a signature with respect to pk_{R}. This leaves a quantum adversary that wants to steal v coins from R, with two options: either he tries to find pk' such that $\mathsf{H}(pk') = h$, or he waits until pk_{R} is revealed by R and computes the corresponding secret key sk_{R}. The first type of attack is believed to be hard because common cryptographic hash functions such as SHA3-512 are known to be preimage resistant even under quantum attacks when appropriately choosing their parameters. While in the second case a quantum adversary can indeed efficiently attack the signature scheme, he has only a very small window of time to carry out this attack[4]. In particular, he has to frontrun the transaction published by R, which is unlikely, assuming that the majority of miners is following the protocol.

A quantum attacker can have more devastating consequences against a deterministic wallet scheme. More concretely, unlike for normal addresses (hashes of public keys), in a deterministic wallet all session keys are related, and in particular efficiently computable from (msk, mpk). Hence, if the adversary manages to learn mpk then he can recover the corresponding master secret key msk and from that recover *all* session secret keys. Hence, all the money that was ever transferred to the cold wallet is at stake.

1.1 Our Contributions

Our main contribution in this work is to give the first construction of a post-quantum secure deterministic wallet. Our scheme is intended to be used on classical computers, and to remain secure even in the presence of quantum adversaries. To achieve our goal we extend the security model of Das et al. [16] to the quantum setting and prove that certain standard post-quantum secure signature schemes can be used to construct post-quantum secure wallets. Concretely, our contributions are as follows:

– We extend the security model for deterministic wallets introduced by Das et al. [16] to the quantum world. In particular, we show that if the underlying signature scheme satisfying the property of honestly rerandomizable keys is post-quantum secure, then it can be used to build post-quantum secure deterministic wallets. We relax the notion of rerandomizable keys as given by [16] to consider only rerandomization of public keys. Subsequently, we show that this relaxed notion is sufficient for the security of wallets, and hence we are able to prove post-quantum security based on this relaxed notion.
– We design the first post-quantum secure signature scheme with rerandomizable public keys. This is achieved by giving a generic construction from a Fiat-Shamir signature scheme based on lattice assumptions.
– We discuss optimizations of our post-quantum secure signature scheme with rerandomizable public keys and evaluate its feasibility for blockchains.

1.2 Our Techniques

Signature schemes with rerandomizable keys [23] are the main building block of the wallet scheme presented in [16]. At a high-level besides the standard algorithms of a digital signature scheme for key generation, signing, and verification, a signature scheme with rerandomizable keys has two additional algorithms, namely `RSig.RandSK` and `RSig.RandPK`. These algorithms take as input the secret key sk, respectively public key pk and randomness ρ, and output fresh keys sk', respectively pk'. Moreover, the unforgeability property of the signature scheme must hold even if the adversary sees signatures that are generated using rerandomized secret keys.

We show that certain post-quantum secure signature schemes support rerandomization of keys and satisfy the security notion of unforgeability under honestly rerandomized keys in the quantum world. In [23] it was shown that Schnorr's signature scheme [39] has rerandomizable keys with unforgeability in the random oracle model (ROM). This motivates to study post-quantum secure Schnorr-like signature schemes. More concretely, we investigate if lattice-based, Schnorr-like signature schemes can have rerandomizable keys with unforgeability in the quantum random oracle model (QROM). Lattice-based schemes are particularly suitable for constructing post-quantum secure rerandomizable signature

[4] In Bitcoin in most cases transactions are considered to be final after 60 minutes.

schemes because (a) lattice-based assumptions are conjectured to be secure under quantum computer attacks; and (b) unlike hash-based signature schemes, lattice-based schemes exhibit an algebraic structure, which enables rerandomization of keys.

The key pair (pk, sk) of such Schnorr-based lattice schemes consists of an instance of a hard lattice problem, where the secret key sk typically follows either the discrete Gaussian distribution or the uniform distribution over a small set. The first idea that comes to mind when rerandomizing keys in the lattice setting is the following: Given (pk, sk) and randomness ρ, sk is rerandomized additively by computing $sk' = sk + \rho$ (as carried out in [23] for Schnorr's scheme). In the lattice setting however, we must ensure that the sum sk' follows the correct distribution, e.g., the Gaussian or uniform distribution. If this is not the case, one can sample a new randomness from ρ in a deterministic way until a valid sk' is generated. Naturally, the same (correct) ρ must be used when rerandomizing pk. This approach satisfies (under a specified distribution of sk) the original definition of signature schemes with rerandomizable keys (see Definition 3), as the initially generated key pair and any rerandomization of it are identically distributed. However, this approach cannot be used for building hot/cold wallets, because the hot and cold wallet must agree on the correct ρ for each session key generation. This contradicts the main goal of using hot/cold wallets, which requires that the cold wallet stays off-line, and hence cannot frequently communicate with the hot wallet to synchronize on ρ. In Appendix D we give more details on this approach as well as others, and argue why they are not suitable in the wallet setting.

In this work we show that the key pair (pk, sk) can still be rerandomized additively in a way that fits to the setting of hot/cold wallets. The main observation that we exploit is that the sum of two Gaussians is also Gaussian distributed (see Lemma 3). Based on this observation, our approach works as follows. Let sk be Gaussian distributed. Given randomness ρ, sk is rerandomized additively by adding to sk a freshly Gaussian distributed secret key sk^*. The key sk^* is deterministically sampled using the randomness ρ, i.e., we use ρ as the randomness required in the Gaussian sampler algorithm. We obtain a rerandomized secret key that is Gaussian distributed, but with a slightly larger standard deviation than the one of the original secret key (cf. Lemma 3). Consequently, we can construct a signature scheme with rerandomizable keys, in which the distribution of rerandomized public keys is computationally indistinguishable to the distribution of the original public key, while rerandomized secret keys follow a different distribution than a freshly sampled secret key. We formally define such relaxed notion in Section 2 and call it a *signature scheme with rerandomizable public keys*. We then show in Section 3 that this notion is sufficient for post-quantum secure wallets and present a lattice-based construction of such a scheme in Section 4 with a security proof in the QROM. Finally, we show in Section 5 that our construction can be instantiated with state-of-the-art lattice-based signature schemes such as qTESLA [7]. Hence, it can use their proposed parameters and enjoy their performance and efficiency.

We emphasize that the post-quantum security model considers the adversary to be quantum while the challenger - representing the honest user in a real-world application - remains classical. As a result, every oracle that is provided by the challenger can be accessed only classically, while oracles that can be accessed by the adversary directly can be accessed using quantum computing power, i.e., in superposition. This describes a threat model where an adversary can use its quantum power to locally access the random oracle, while he observes signatures created by a user on a classical machine. In our work, we consider this standard post-quantum security model since it entails that the cryptographic scheme is still used on classical computers. This is, in contrast to the (fully-) quantum setting, where the scheme itself is implemented on quantum computers as well. While this is a stronger security model, it is more of theoretical interest as it requires users to have access to quantum computers as well.

1.3 Related Work

The concept of hot/cold wallets is used in many cryptocurrencies in order to provide stronger security guarantees to its users. Various works have proposed formal security models for analyzing the security of wallet schemes. Gutoski and Stebila [25] have discussed flaws in the BIP32 construction [3] and possible countermeasures. However, they do not consider the standard notion of unforgeability but rather a restricted model where the adversary can corrupt the cold wallet and recover secret keys. Other works worth mentioning are "privilege escalation attacks" by Fan et al. [22] which unfortunately lacks any formal security analysis, and the analysis of the Bitcoin Electrum wallet in the Dolev-Yao model by Turuani et al. [42]. The latter considers cryptographic primitives (e.g., signature schemes and encryption schemes) as idealized objects, hence fails to capture potential vulnerabilities such as related key attacks which are relevant in case of hot/cold wallets.

As mentioned earlier, we closely follow the model introduced by Das et al. [16], where the notion of a *stateful deterministic wallet* is introduced and two desirable security properties called *wallet unlinkability* and *wallet unforgeability* are considered. The first property ensures that the session public keys generated by SW.RandPK are unlinkable to the master public key. This property is guaranteed as long as the hot wallet has not been corrupted. The second property ensures unforgeability of signatures signed by the secret keys of the cold wallet even when the hot wallet is corrupted.

According to [16] a *stateful deterministic wallet* SW consists of two components – a hot wallet and a cold wallet which share a common state St. SW is given by a tuple of algorithms (SW.KGen, SW.RandSK, SW.RandPK, SW.Sign, SW.Verify), where the session public key and secret key derivation algorithms SW.RandPK and SW.RandSK are run respectively within the hot and cold wallet to deterministically derive matching session (public/secret) keys from the (public/secret) master keys. Unlike deterministic wallets in use (e.g., the BIP32 construction), the state St of the wallet scheme of [16] is refreshed within the (hot/cold) wallets with each key derivation. This approach allows to show forward unlinkability, which intuitively means that even upon leakage of the state, all session keys derived *before* the state leakage remain unlinkable. The second security property – *wallet unforgeability* – is achieved by a reduction to standard EUF-CMA security of a concrete signature scheme (such as ECDSA and Schnorr) in a modularized fashion. As the intermediary step the authors show that these signature schemes satisfy the properties of signature schemes with rerandomizable keys.

We note that the model by Das et al. [16] only considers adversaries in the classical setting and does not protect against quantum adversaries. Our work fills this gap by designing the first *post-quantum secure deterministic wallet.*

Many prior works have investigated lattice-based Fiat-Shamir signatures, e.g., [7, 9, 18, 19, 30], and in particular, their security was analyzed in the QROM, e.g., by [17, 27, 29, 44]. To the best of our knowledge we propose the first work on lattice-based signature schemes with honestly rerandomizable keys and prove its security in the QROM. Inspired by Das et al. [16] and Fleischhacker et al. [23], we use the abstraction of signature schemes with rerandomizable keys but transfer this concept to the post-quantum setting. We include further related work in Appendix A.

2 Preliminaries

We let $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ denote the set of natural numbers, integers, and real numbers, respectively. For any positive integer k we write $[k]$ to denote the set of integers $\{1, \dots, k\}$. For a positive integer q we let $\mathbb{Z}_q$ denote the set of integers in the range $[-\frac{q}{2}, \frac{q}{2}) \cap \mathbb{Z}$. We define the ring $R = \mathbb{Z}[x]/\langle x^n + 1 \rangle$ and its quotient $R_q = R/qR$, where n is a power of 2. Elements in R and R_q (including $\mathbb{Z}$ and $\mathbb{Z}_q$) are denoted by regular font letters. Column vectors and matrices with entries from R or R_q are denoted by bold lower-case letters and bold upper-case letters, respectively. We define the ℓ_2 and ℓ_∞ norms of $v = \sum_{i=0}^{n-1} v_i x^i \in R$ by $\|v\| = (\sum_{i=0}^{n-1} |v_i|^2)^{1/2}$ and $\|v\|_\infty = \max_i |v_i|$, respectively. For $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$ we define $\|\mathbf{w}\| = (\sum_{i=1}^{k} \|w_i\|^2)^{1/2}$ and $\|\mathbf{w}\|_\infty = \max_i \|w_i\|_\infty$. We let $\mathbb{T}_\kappa^n$ denote the set of all $(n-1)$-degree polynomials with coefficients from $\{-1, 0, 1\}$ and Hamming weight κ. We always denote the security parameter by $\lambda \in \mathbb{N}$, and $o(\lambda)$ denotes a linear function in λ. A function $f : \mathbb{N} \longrightarrow \mathbb{R}$ is called *negligible* if there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, it holds $f(n) < \frac{1}{p(n)}$ for any polynomial p. With $\mathrm{negl}(\lambda)$ we denote a negligible function in λ. A probability is called overwhelming if it is at least $1 - \mathrm{negl}(\lambda)$. The *statistical distance* between two distributions X, Y over a countable domain D is defined by $\frac{1}{2} \sum_{n \in D} |X(n) - Y(n)|$. We write $x \leftarrow D$ to denote that x is sampled according to a distribution D. We let $x \leftarrow_\$ S$ denote choosing x uniformly random from a finite set S. Unless specified otherwise, every adversary is considered to be an efficient quantum polynomial time algorithm.

2.1 Quantum Random Oracle Model

In this section, we recall the quantum random oracle model and existing results that we will use. Since quantum computation is only necessary in the proofs of these results, we do not provide information on quantum computation here, but refer to [37] for a detailed discussion on the topic.

In [10], Bellare and Rogaway introduced the *random oracle model* (ROM). In this model every party has access to an oracle implementing a random function. Upon being queried on some input x, the oracle answers with a random output y. Every further invocation on input x, even by other parties, results in the same y. In security proofs, one often models a hash function as a random oracle. Since hash functions are public, Boneh et al. [13] observed that the ROM is not appropriate in the post-quantum setting. In the real world an adversary equipped with a quantum computer is able to implement the hash function and evaluate it in superposition. Thus, Boneh et al. introduced the *quantum random oracle model* (QROM). In this model, parties with quantum computing power get access to the oracle $|\mathsf{H}\rangle$, where $|\mathsf{H}\rangle : |x, y\rangle \mapsto |x, y \oplus \mathsf{H}(x)\rangle$. In our proofs we will also consider reprogrammed random oracles. For a random oracle H we write $\mathsf{H}_{x \to y}$ for the random oracle that is reprogrammed on input x to y. Further on, we denote the classical random oracle by the symbol H and the quantum random oracle by the notation $|\mathsf{H}\rangle$.

Nowadays, the QROM is considered the de facto standard for post-quantum security proofs of cryptographic primitives which rely on random oracles. Below we describe some results for quantum random oracles that are required for our proofs.

The one-way to hiding (O2H) lemma [43] is an important tool for security proofs in the quantum random oracle model. It gives bounds on the advantage of an adversary in distinguishing between different random oracles when the adversary is allowed to query them in superposition. Below we state the lemma using the reformulation by Ambainis et al. [8].

Lemma 1 (One-way to hiding (O2H) [8]). *Let G, $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be random functions, let z be a random value, and let $\mathcal{S} \subset \mathcal{X}$ be a random set such that $\forall x \notin \mathcal{S}$, $\mathsf{G}(x) = \mathsf{H}(x)$. $(\mathsf{G}, \mathsf{H}, \mathcal{S}, z)$ may have arbitrary joint distribution. Furthermore, let $\mathcal{A}^{|\mathsf{H}\rangle}$ be a quantum oracle algorithm which queries $|\mathsf{H}\rangle$ at most q times. Let Ev be an arbitrary classical event. Define an oracle algorithm $\mathcal{B}^{|\mathsf{H}\rangle}$ as follows: Pick $i \leftarrow_\$ [q]$. Run $\mathcal{A}^{|\mathsf{H}\rangle}(z)$ until just before its i-th round of queries to $|\mathsf{H}\rangle$. Measure the query in the computational basis, and output the measurement outcome. It holds that*

$$\left|\Pr[\mathsf{Ev} : \mathcal{A}^{|\mathsf{H}\rangle}(z)] - \Pr[\mathsf{Ev} : \mathcal{A}^{|\mathsf{G}\rangle}(z)]\right| \leq 2q\sqrt{\Pr[x \in \mathcal{S} : \mathcal{B}^{|\mathsf{H}\rangle}(z) \Rightarrow x]}\,.$$

Another tool that we will use are Zhandry's small range distributions, defined below. These are distributions where the set of possible outputs is limited.

Definition 1 (Small-range distributions [45]). *Let $\mathcal{X}$, $\mathcal{Y}$ be sets, r be an integer, D be a distribution on $\mathcal{Y}$, P be a random function from $\mathcal{X}$ to $[r]$, and $\vec{y} = (y_1, \ldots, y_r)$ be r samples of D. Define a function $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$ by $\mathsf{H}(x) \mapsto y_{P(x)}$. The distribution of H, induced by P and $\vec{y}$, is called a* small-range distribution *with r samples of D.*

The following lemma provides a bound on the distinguishing advantage between a random oracle and an oracle drawn from a small-range distribution when superposition access is granted.

Lemma 2 ([45]). *There is a universal constant C such that, for any set $\mathcal{X}$ and $\mathcal{Y}$, distribution D on $\mathcal{Y}$, integer l, and any quantum algorithm $\mathcal{A}$ making q queries to an oracle $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$, the following two cases are indistinguishable, except with probability less than $\frac{Cq^3}{l}$:*

- *$\mathsf{H}(x) = y_x$ where $\vec{y}$ is a list of samples of D of size $|\mathcal{X}|$.*
- *H is drawn from the small-range distribution with l samples of D.*

2.2 Cryptographic Primitives

Definition 2 (Signature Scheme). *Let λ be a security parameter. A signature scheme Sig with key space $\mathcal{K}$, message space $\mathcal{M}$, and signature space $\mathcal{S}$ is a tuple of polynomial-time algorithms $(\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ such that*

$\mathtt{KGen}(1^\lambda)$ *is the key generation algorithm that outputs a key pair $(pk, sk) \in \mathcal{K}$, where pk is a public key and sk is a secret key.*

$\mathtt{Sign}(sk, m)$ *is the signing algorithm that takes as input a secret key sk and a message $m \in \mathcal{M}$. It outputs a signature $\sigma \in \mathcal{S}$.*

$\mathtt{Verify}(pk, m, \sigma)$ *is the verification algorithm that takes as input a public key pk, a message m with a signature σ. It outputs 1 if σ is valid and 0 otherwise.*

Correctness: *A signature scheme is correct if for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(pk, sk) \leftarrow \mathtt{KGen}(1^\lambda)$, and all $\sigma \leftarrow \mathtt{Sign}(sk, m)$, it holds that $\Pr[\mathtt{Verify}(pk, m, \mathtt{Sign}(sk, m)) = 1] \geq 1 - \mathrm{negl}(\lambda)$.*

We will use the notion of signature schemes with rerandomizable keys [23]. In the following we recall its definition.

Definition 3 (Signature Scheme with Rerandomizable Keys). *A signature scheme with perfectly rerandomizable keys $\mathtt{RSig}$ is given by a tuple of algorithms:*

$$(\mathtt{RSig.KGen}, \mathtt{RSig.RandSK}, \mathtt{RSig.RandPK}, \mathtt{RSig.Sign}, \mathtt{RSig.Verify}),$$

where $\mathtt{RSig.KGen}$, $\mathtt{RSig.Sign}$, $\mathtt{RSig.Verify}$ satisfy the definition of a standard signature scheme (cf. Definition 2). For randomness space $\mathcal{R}$, $(\mathtt{RSig.RandSK}, \mathtt{RSig.RandPK})$ are two polynomial-time algorithms such that

$\mathtt{RSig.RandSK}(sk, \rho)$ *is a secret key rerandomization algorithm that takes as input the secret key sk and a randomness $\rho \in \mathcal{R}$ and outputs a randomized secret key sk'.*

$\mathtt{RSig.RandPK}(pk, \rho)$ *is a public key rerandomization algorithm that takes as input the public key pk and a randomness $\rho \in \mathcal{R}$ and outputs a randomized public key pk'.*

$\mathtt{RSig}$ *satisfies the following properties:*

```
Game EUF-CMA$^{\mathcal{A}}_{\Sigma}$
1: $\mathcal{Q} := \varnothing$
2: $H := \varnothing$
3: $(pk, sk) \leftarrow \texttt{KGen}(1^\lambda)$
4: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{O}}(pk)$
5: if $(m^* \in \mathcal{Q})$ then
6:     return 0
7: return $\texttt{Verify}(pk, m^*, \sigma^*)$

$\mathsf{H}(m')$
1: if $(m' \in H)$ then
2:     return $\mathsf{H}(m') \in H$
3: $\mathsf{H}(m') \leftarrow_\$ \{0,1\}^{o(\lambda)}$
4: $H := H \cup \{(m', \mathsf{H}(m'))\}$
5: return $\mathsf{H}(m')$

$\mathsf{O}(m)$
1: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2: $\sigma \leftarrow \texttt{Sign}(sk, m)$
3: return $\sigma$
```

Fig. 1. The security game EUF-CMA of signature schemes.

Rerandomizability of keys: *For all $\lambda \in \mathbb{N}$, all $(sk, pk) \in \texttt{RSig.KGen}(1^\lambda)$, and all $\rho \in \mathcal{R}$, the distributions of (sk', pk') and (sk'', pk'') are identical, where $(sk'', pk'') \leftarrow \texttt{RSig.KGen}(1^\lambda)$ and $sk' \leftarrow \texttt{RSig.RandSK}(sk, \rho)$, $pk' \leftarrow \texttt{RSig.RandPK}(pk, \rho)$.*

Correctness under rerandomizable keys:

1. *For all $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \texttt{RSig.KGen}(1^\lambda)$, $m \in \mathcal{M}$, and all $\sigma \leftarrow \texttt{RSig.Sign}(sk, m)$, it holds that*

$$\Pr[\texttt{RSig.Verify}(pk, m, \texttt{RSig.Sign}(sk, m)) = 1] \geq 1 - \text{negl}(\lambda)\,.$$

2. *For all $(pk, sk) \leftarrow \texttt{RSig.KGen}(1^\lambda)$, all $\rho \in \mathcal{R}$, $m \in \mathcal{M}$, and for a pair of rerandomized keys $sk' \leftarrow \texttt{RSig.RandSK}(sk, \rho)$ and $pk' \leftarrow \texttt{RSig.RandPK}(pk, \rho)$, it holds*

$$\Pr[\texttt{RSig.Verify}(pk', m, \texttt{RSig.Sign}(sk', m)) = 1] \geq 1 - \text{negl}(\lambda)\,.$$

We also consider a relaxed version of Definition 3. In the following definition we introduce the notion of *signature schemes under rerandomizable public keys*, where the distribution of rerandomized public keys is computationally indistinguishable from the distribution of the original public key, but where the same does not hold for secret keys. We present a concrete instantiation of such a scheme in Section 4.

Definition 4 (Signature Scheme with Rerandomizable Public Keys). *A signature scheme with perfectly rerandomizable public keys $\texttt{RSig}'$ is given by a tuple of algorithms $(\texttt{RSig}'.\texttt{KGen}, \texttt{RSig}'.\texttt{RandSK}, \texttt{RSig}'.\texttt{RandPK}, \texttt{RSig}'.\texttt{Sign}, \texttt{RSig}'.\texttt{Verify})$, which are defined as in Definition 3. $\texttt{RSig}'$ satisfies the following properties:*

Rerandomizability of public keys: *For all $\lambda \in \mathbb{N}$, all public keys $(\cdot, pk) \leftarrow \texttt{RSig}'.\texttt{KGen}(1^\lambda)$ and $\rho \in \mathcal{R}$, the distributions of pk' and pk'' are computationally indistinguishable, where $pk' \leftarrow \texttt{RSig}'.\texttt{RandPK}(pk, \rho)$, and $pk'' \leftarrow \texttt{RSig}'.\texttt{KGen}(1^\lambda)$.*

Correctness under rerandomizable keys: *This property is defined as the property of correctness for signature schemes under rerandomizable keys in Definition 3.*

Simulatability: *For all $\lambda \in \mathbb{N}$, all $(sk, pk) \leftarrow \texttt{RSig}'.\texttt{KGen}(1^\lambda)$, and all $m \in \mathcal{M}$, there exists a polynomial-time algorithm $\mathcal{T}$ which on input pk and m outputs a signature $\sigma \in \mathcal{S}$. It must hold that for $\kappa \in \text{poly}(\lambda)$ the distributions $\{\sigma_1, \cdots, \sigma_\kappa\}$ and $\{\sigma'_1, \cdots, \sigma'_\kappa\}$ are computationally indistinguishable where $\sigma_i \leftarrow \mathcal{T}(pk, m)$ and $\sigma'_i \leftarrow \texttt{RSig}'.\texttt{Sign}(sk, m)$ for $i \in [\kappa]$.*

2.3 Security Notions

Security of signature schemes is captured by the standard security notion of *existential unforgeability under adaptive chosen-message attacks* (EUF-CMA), presented below.

```
Game EUF-CMA-HRK$^{\mathcal{A}}_{\mathtt{RSig}}$
 1: $\mathsf{RList} := \varnothing$
 2: $\mathcal{Q} := \varnothing$
 3: $H := \varnothing$
 4: $(pk, sk) \leftarrow \mathtt{RSig.KGen}(1^\lambda)$
 5: $(m^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{Rand},\mathsf{OHR}}(pk)$
 6: if $(m^* \in \mathcal{Q})$ then
 7:     return 0
 8: if $(\rho^* \neq \mathsf{NULL})$ then
 9:     if $(\rho^* \notin \mathsf{RList})$ then
10:         return 0
11:     $pk \leftarrow \mathtt{RSig.RandPK}(pk, \rho^*)$
12: return $\mathtt{RSig.Verify}(pk, m^*, \sigma^*)$

$\mathsf{H}(m')$ (see Figure 1)

Rand
 1: $\rho \leftarrow_\$ \mathcal{R}$
 2: $\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
 3: return $\rho$

$\mathsf{OHR}(m, \rho)$
 1: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
 2: if $(\rho \neq \mathsf{NULL})$ then
 3:     if $(\rho \notin \mathsf{RList})$ then
 4:         return $\perp$
 5:     $sk \leftarrow \mathtt{RSig.RandSK}(sk, \rho)$
 6: $\sigma \leftarrow \mathtt{RSig.Sign}(sk, m)$
 7: return $\sigma$
```

Fig. 2. The security game EUF-CMA-HRK of signature schemes with rerandomizable (public) keys.

Definition 5 (EUF-CMA Security). *Let $H : \{0,1\}^* \to \{0,1\}^{o(\lambda)}$ be a hash function modeled as (quantum) random oracle. A signature scheme Σ is called $(t, q_{\mathtt{Sign}}, q_H, \varepsilon)$-EUF-CMA in the (quantum) random oracle model if for any adversary A running in time at most t and making at most $q_{\mathtt{Sign}}$ signature queries and at most q_H (superposition) queries to H, the game EUF-CMA$^{\mathcal{A}}_{\Sigma}$ depicted in Figure 1 outputs 1 with probability at most ε, i.e.,* $\Pr[\textit{EUF-CMA}^{\mathcal{A}}_{\Sigma} = 1] \leq \varepsilon$.

In the following we present the notion of *EUF-CMA-HRK security under honestly rerandomizable keys* due to [16]. This notion is similar to *EUF-CMA-RK security under rerandomizable keys* due to [23], however with certain differences which makes it a weaker notion. In the EUF-CMA-RK game, an adversary $\mathcal{A}$ gets access to a signing oracle. The signing oracle takes a message and a randomness as input and provides a signature on this message under the rerandomized key as an answer. Note that the rerandomized key was derived from the randomness input by $\mathcal{A}$. This means that $\mathcal{A}$ can obtain signatures under keys with randomness of $\mathcal{A}$'s choice. $\mathcal{A}$ can win the EUF-CMA-RK game if it can produce a valid forgery under a rerandomized key of its choice (note that the randomness can also be null).

In the EUF-CMA-HRK game, we restrict $\mathcal{A}$'s capabilities in the following way. In addition to the signing oracle, in the EUF-CMA-HRK game $\mathcal{A}$ is given access to a Rand oracle to derive a fresh randomness. This randomness can be later used to get a signature under the rerandomized key by querying the signing oracle. Here, $\mathcal{A}$ can only win the EUF-CMA-HRK game if it produces a valid forgery under a rerandomized key, where the underlying randomness was obtained honestly by querying the Rand oracle. We formally present *EUF-CMA-HRK security under honestly rerandomizable (public) keys* below.

Definition 6 (EUF-CMA-HRK Security under Honestly Rerandomized (Public) Keys). *Let $H : \{0,1\}^* \to \{0,1\}^{o(\lambda)}$ be a hash function modeled as (quantum) random oracle. A signature scheme with honestly rerandomizable (public) keys* RSig *is called $(t, q_{\mathtt{Sign}}, q_H, \varepsilon)$-EUF-CMA-HRK in the (quantum) random oracle model if for any adversary A running in time at most t and making at most $q_{\mathtt{Sign}}$ signature queries and at most q_H (quantum) random oracle queries to H, the game EUF-CMA-HRK$^{\mathcal{A}}_{\mathtt{RSig}}$ depicted in Figure 2 outputs 1 with probability at most ε, i.e.,* $\Pr[\textit{EUF-CMA-HRK}^{\mathcal{A}}_{\mathtt{RSig}} = 1] \leq \varepsilon$.

```
$\boxed{\mathsf{LB}.\mathtt{KGen}(1^\lambda)}$
1: $\mathbf{s} \leftarrow \chi^{k_2}$, $\mathbf{e} \leftarrow \chi^{k_1}$
2: $\mathbf{b} \leftarrow \mathbf{As} + \mathbf{e} \pmod q$
3: $sk := (\mathbf{s}, \mathbf{e})$, $pk := \mathbf{b}$
4: **return** $(sk, pk)$

$\boxed{\mathsf{LB}.\mathtt{Verify}(pk, m, (\mathbf{z}_1, \mathbf{z}_2, c))}$
1: $\mathbf{w} \leftarrow \mathbf{Az}_1 + \mathbf{z}_2 - \mathbf{b}c \pmod q$
2: **if** $\big((\mathbf{z}_1, \mathbf{z}_2) \in R_{B_1}^{k_2} \times R_{B_2}^{k_1} \wedge$
       $\mathsf{H}(\mathbf{w}, m) = c\big)$ **then**
3:     **return** 1
4: **return** 0

$\boxed{\mathsf{LB}.\mathtt{Sign}(sk, m)}$
1: $\mathbf{r} \leftarrow_\$ \{0,1\}^{o(\lambda)}$
2: $\mathsf{ctr} \leftarrow 1$
3: $(\mathbf{y}_1, \mathbf{y}_2) \in R_Y^{k_2} \times R_Y^{k_1} \leftarrow \mathsf{E}(\mathbf{r}, \mathsf{ctr})$
4: $\mathbf{v} \leftarrow \mathbf{Ay}_1 + \mathbf{y}_2 \pmod q$
5: $c \leftarrow \mathsf{H}(\mathbf{v}, m)$
6: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + \mathbf{s}c$
7: $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + \mathbf{e}c$
8: **if** $\big((\mathbf{z}_1, \mathbf{z}_2) \notin R_{B_1}^{k_2} \times R_{B_2}^{k_1}\big)$ **then**
9:     $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
10:    goto 3
11: **return** $(\mathbf{z}_1, \mathbf{z}_2, c)$
```

Fig. 3. A formal description of a generic (non-optimized) Fiat-Shamir signature scheme from lattice assumptions.

2.4 Lattice-Based Fiat-Shamir Signatures

In this section we review a generic construction of lattice-based Fiat-Shamir signatures. We first define the discrete Gaussian distribution and recall a lemma, which shows that the sum of Gaussian distributed random variables is also Gaussian distributed. This property is crucial for our analysis.

Definition 7 (Discrete Gaussian Distribution). *The discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma,\mathbf{c}}$ over $\mathbb{Z}^n$ with standard deviation $\sigma > 0$ and center $\mathbf{c} \in \mathbb{R}^n$ is defined as follows: For every $\mathbf{x} \in \mathbb{Z}^n$ the probability of $\mathbf{x}$ is given by $D_{\mathbb{Z}^n,\sigma,\mathbf{c}}(\mathbf{x}) = \rho_{\sigma,\mathbf{c}}(\mathbf{x})/\rho_{\sigma,\mathbf{c}}(\mathbb{Z}^n)$, where $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(\frac{-\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2})$ and $\rho_{\sigma,\mathbf{c}}(\mathbb{Z}^n) = \sum_{\mathbf{x}\in\mathbb{Z}^n} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. The subscript $\mathbf{c}$ is taken to be $\mathbf{0}$ when omitted.*

Lemma 3 ([14, Theorem 9]). *Let $\mathcal{L} \subseteq \mathbb{Z}^{\Updownarrow}$ be a lattice and $\sigma \in \mathbb{R}$. For $i = 1, \ldots, n$ let $\mathbf{t}_i \in \mathbb{Z}^m$ and let X_i be mutually independent random variables sampled from $D_{\mathcal{L}+\mathbf{t}_\rangle,\sigma}$. Let $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{Z}^n$ and define $d = \gcd(c_1, \ldots, c_n)$, $\mathbf{t} = \sum_1^n c_i \mathbf{t}_i$. Suppose that $\sigma > \|\mathbf{c}\| \cdot \eta_\varepsilon(\mathcal{L})$, where $\eta_\varepsilon(\mathcal{L})$ is the smoothing parameter [33] for some negligible ε. Then $Z = \sum_1^n c_i X_i$ is statistically close to $D_{d\mathcal{L}+\mathbf{t},\|\mathbf{c}\|\sigma}$.*

Next, we describe two functions used in the signature scheme:
(1) $\mathsf{E} : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a function that expands given strings to any desired length. It is used to extract the randomness used for signing, and (2) $\mathsf{H} : \{0,1\}^* \longrightarrow \mathbb{T}_\kappa^n$ is a hash function modeled as a (quantum) random oracle and used for signing and verification.

The signature scheme is formally described in Figure 3. It makes use of a uniformly random matrix $\mathbf{A} \in R_q^{k_1 \times k_2}$, which is publicly known and shared among all users in a multi-user setting. We assume that $\mathbf{A}$ is an implicit input to all algorithms of the scheme in addition to all algorithms in Section 4. In order to save bandwidth it can also be generated by expanding a uniformly random seed using the function E, and including the seed in the secret and public key rather than storing the whole matrix $\mathbf{A}$. In this case, E is modelled as a random oracle. We note that this setting makes sense in the context of blockchains, since the randomly chosen seed can be included in the first block known as the genesis block, which is assumed to be honestly generated. Furthermore, since $\mathbf{A}$ is computed as the output of the random oracle on input the seed, $\mathbf{A}$ is truly random and cannot have a trapdoor embedded as shown in [32].

Basically, the key generation algorithm generates an instance of a computationally hard lattice problem called *Module Learning with Errors* (MLWE) [28] (or a special variant of it such as Ring Learning with

Errors (RLWE) [31]). The secret of this instance is chosen from some distribution χ. In the state-of-the-art lattice-based signature schemes, e.g., Dilithium [19] and qTESLA [7], the distribution of the secrets is either the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ or the distribution R_d that outputs uniformly random polynomials from R whose ℓ_∞ norm is bounded by some integer $d \geq 1$.

A signature consists of a tuple $(\mathbf{z}_1, \mathbf{z}_2, c)$, where the pair $(\mathbf{z}_1, \mathbf{z}_2)$ is uniformly random over a subset of $R^{k_2} \times R^{k_1}$ and $c \in \mathbb{T}^n_\kappa$ is output from the random oracle H. The vectors $\mathbf{z}_1, \mathbf{z}_2$ are each generated by adding a masking term to a term related to the secret key and c. More precisely, we have $\mathbf{z}_1 = \mathbf{y}_1 + \mathbf{s}c$ and $\mathbf{z}_2 = \mathbf{y}_2 + \mathbf{e}c$, where the secret masking pair $(\mathbf{y}_1, \mathbf{y}_2)$ is uniformly random over $R_Y^{k_2} \times R_Y^{k_1}$ and $R_Y \subset R$ for some predefined positive integer Y. The signature is only output after verifying that the pair $(\mathbf{z}_1, \mathbf{z}_2)$ lies in $R_{B_1}^{k_2} \times R_{B_2}^{k_1}$, i.e., $\|\mathbf{z}_1\|_\infty \leq B_1$ and $\|\mathbf{z}_2\|_\infty \leq B_2$, where the bounds B_1, B_2 are defined depending on the distribution of the secret key. This ensures that signatures are uniformly distributed over $R_{B_1}^{k_2} \times R_{B_2}^{k_1} \times \mathbb{T}^n_\kappa$ and do not leak information about the secret key. If this is not the case, the algorithm restarts with a fresh masking pair $(\mathbf{y}_1, \mathbf{y}_2)$. The average number of repetitions is denoted by $M = O(1)$. Valid signatures are generated with probability $\left(\frac{2B_1+1}{2Y+1}\right)^{k_2 n} \cdot \left(\frac{2B_2+1}{2Y+1}\right)^{k_1 n}$, which is usually chosen such that it is at least $1/M$. We note that this generic construction can be optimized by either following the technique due to Bai and Galbraith [9] (adopted in qTESLA) or the approach used in Dilithium. The first one optimizes the signature size, while the second one optimizes the total size of public key and signature.

Finally, the EUF-CMA security of lattice-based Fiat-Shamir signatures in the quantum random oracle model was analyzed in several works, e.g., in [7, 17, 19, 27, 29, 44].

3 The Stateful Model for Wallets

Our formal security model for post-quantum secure stateful deterministic wallets is based on the model of [16]. In this section, we recall the formal definition of a stateful wallet and the security properties that we want to guarantee for such a wallet. A stateful deterministic wallet scheme consists of two entities, a cold wallet and a hot wallet, that can respectively derive a valid pair of secret and public keys without the need for any interaction among each other. In more detail, upon initialization of the scheme, the cold wallet generates a master key pair (msk, mpk) and some initial state information St_0 and forwards (mpk, St_0) to the hot wallet. After this initial setup, the idea is that an arbitrary number of valid session key pairs can be generated by using the session secret/public key derivation algorithms within the respective wallets without further interaction. More precisely the public key derivation algorithm takes as input the current state and the master public key to generate a session public key. While the secret key derivation takes as inputs the current state and the master secret key and generates a session secret key. Since both public key and secret key derivation algorithms are deterministic, and the two wallets share the same current state, the key derivation algorithms output a valid session key pair. In order to keep track of which key has been derived with which state, each session key is indexed by a parameter ID, which is given as input into the key derivation procedures. In the following we recall the definition of a deterministic stateful wallet scheme and its correctness.

Definition 8 (Stateful Wallet). *A* stateful wallet scheme *is a tuple of algorithms* $\texttt{SW} := (\texttt{SW.KGen}, \texttt{SW.RandSK}, \texttt{SW.RandPK}, \texttt{SW.Sign}, \texttt{SW.Verify})$, *which are defined as follows:*

$\underline{\texttt{SW.KGen}}$**:** *The master key generation algorithm takes as input public parameters param and outputs a master key pair (msk, mpk) as well as an initial state St_0.*

$\underline{\texttt{SW.RandSK}}$**:** *The secret key derivation algorithm takes as input a master secret key msk, a state St and an identity ID and outputs a session secret key sk_{ID} and the state St.*

```
$\boxed{\texttt{SW.KGen}(1^\lambda)}$
  1: $St \leftarrow_\$ \{0,1\}^\lambda$
  2: $(mpk, msk) \leftarrow \texttt{RSig.KGen}(1^\lambda)$
  3: return $(St, mpk, msk)$
$\boxed{\texttt{SW.Sign}(sk, pk, m)}$
  1: $m' \leftarrow (m, pk)$
  2: $\sigma \leftarrow \texttt{RSig.Sign}(sk, m')$
  3: return $\sigma$
$\boxed{\texttt{SW.Verify}(pk, m, \sigma)}$
  1: $m' \leftarrow (m, pk)$
  2: return $\texttt{RSig.Verify}(pk, m', \sigma)$

$\boxed{\texttt{SW.RandSK}(msk, ID, St)}$
  1: $(\rho, St) \leftarrow \mathsf{H}(St, ID)$
  2: $sk_{ID} \leftarrow \texttt{RSig.RandSK}(msk, \rho)$
  3: return $(sk_{ID}, St)$
$\boxed{\texttt{SW.RandPK}(mpk, ID, St)}$
  1: $(\rho, St) \leftarrow \mathsf{H}(St, ID)$
  2: $pk_{ID} \leftarrow \texttt{RSig.RandPK}(mpk, \rho)$
  3: return $(pk_{ID}, St)$
```

Fig. 4. Generic Construction of a stateful deterministic wallet SW from a signature scheme with honestly rerandomizable keys RSig and a random oracle H.

SW.RandPK: *The public key derivation algorithm takes as input a master public key mpk, a state St and an identity ID and outputs a session secret key pk_{ID} and the state St.*

SW.Sign: *The probabilistic signing algorithm takes as input a session secret key sk_{ID} for some ID and a message m and outputs a signature σ.*

SW.Verify: *The verification algorithm takes as input a session public key pk_{ID} for some ID, a message m, and a signature σ and outputs 1 if σ is a valid signature for m under public key pk_{ID}. It outputs 0 otherwise.*

Definition 9 (Correctness of Stateful Wallets). *For $n \in \mathbb{N}$, any $(St_0, msk, mpk) \in \texttt{SW.KGen}(param)$, and any $\vec{ID} := (ID_1, ..., ID_n) \in \{0,1\}^*$, we define the sequence (sk_i, St_i) and (pk_i, St_i) for $1 \leq i \leq n$ recursively as*

$$(sk_i, St_i) := \texttt{SW.RandSK}(msk, ID_i, St_{i-1}),$$
$$(pk_i, St_i) := \texttt{SW.RandPK}(mpk, ID_i, St_{i-1}).$$

SW *is* correct *if for all $m \in \{0,1\}^*$ and i with $1 \leq i \leq n$ it holds that*

$$\Pr_{\sigma \leftarrow\$ \texttt{SW.Sign}(sk_i, m)}[\texttt{SW.Verify}(pk_i, \sigma, m) = 1] \geq 1 - \text{negl}(\lambda)\,.$$

A generic construction of a stateful deterministic wallet scheme SW := (SW.KGen, SW.RandSK, SW.RandPK, SW.Sign, SW.Verify) from a signature scheme with honestly rerandomizable keys RSig following Definition 8 is presented in Figure 4. Such a scheme should satisfy the following two security properties - *wallet unlinkability* and *wallet unforgeability* - which are described below.

3.1 Wallet Unlinkability

Intuitively, the unlinkability property guarantees that session public keys that have been derived from the same master public key are computationally indistinguishable from the distribution of session public keys that have been derived from a different, independently chosen master public key. However, considering that hot wallet corruptions reveal the state and hence trivially break the unlinkability property, [16] introduces the notion of forward unlinkability. This notion guarantees unlinkability prior to any hot wallet corruption.

```
Game WUNL
1:  $(mpk, msk, St) \leftarrow \texttt{SW.KGen}()$
2:  $b \leftarrow_\$ \{0,1\}$
3:  $ID^* \leftarrow \mathcal{A}_1^{\mathsf{WalSign},\mathsf{PK}}(mpk)$
4:  if $(Keys[ID^*] \neq \bot)$ then
5:      return 0
6:  $(pk^0_{ID^*}, St^*) \leftarrow \texttt{SW.RandPK}(mpk, ID^*, St)$
7:  $(sk^0_{ID^*}, St^*) \leftarrow \texttt{SW.RandSK}(msk, ID^*, St)$
8:  $St \leftarrow St^*$
9:  $(\overline{mpk}, \overline{msk}, \overline{St}) \leftarrow \texttt{SW.KGen}()$
10: $(pk^1_{ID^*}, St^*) \leftarrow \texttt{SW.RandPK}(\overline{mpk}, ID^*, \overline{St})$
11: $(sk^1_{ID^*}, St^*) \leftarrow \texttt{SW.RandSK}(\overline{msk}, ID^*, \overline{St})$
12: $Keys[ID^*] \leftarrow (pk^b_{ID^*}, sk^b_{ID^*})$
13: $b' \leftarrow \mathcal{A}_2^{\mathsf{WalSign},\mathsf{PK},\mathsf{getState}}(mpk, pk^b_{ID^*})$
14: return $b' = b$
```

```
Oracle PK(ID)
1: $(pk_{ID}, St^*) \leftarrow \texttt{SW.RandPK}(mpk, ID, St)$
2: $(sk_{ID}, St^*) \leftarrow \texttt{SW.RandSK}(msk, ID, St)$
3: $Keys[ID] \leftarrow (pk_{ID}, sk_{ID})$
4: $St \leftarrow St^*$
5: return $pk_{ID}$
```

```
Oracle getState()
1: return $St$
```

```
Oracle WalSign(m, ID)
1: if $(Keys[ID] = \bot)$ then
2:     return $\bot$
3: $(pk_{ID}, sk_{ID}) \leftarrow Keys[ID]$
4: $\sigma \leftarrow \texttt{SW.Sign}(sk_{ID}, m)$
5: return $\sigma$
```

Fig. 5. Unlinkability game WUNL for stateful wallets.

The formal security game for unlinkability is defined in Figure 5 and proceeds as follows: Upon the initialization of the wallet scheme by executing $(mpk, msk, St) \leftarrow_\$ \texttt{SW.KGen}()$, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ obtains mpk and runs its subprocedure $\mathcal{A}_1$ on input mpk, where $\mathcal{A}_1$ has access to oracles WalSign and PK. These oracles represent the adversary's capability to observe signatures with corresponding session public keys of the wallet on the ledger. More concretely, $\mathcal{A}_1$ can call WalSign on an arbitrary message m and any ID and receives a valid signature for m under public key pk_{ID}. Further, $\mathcal{A}_1$ can query the PK oracle on any ID and receives the session public key pk_{ID}.

Finally, $\mathcal{A}_1$ outputs an ID^*. If neither WalSign nor PK has been queried on ID^* before, the game proceeds to the challenge phase, in which two session key pairs $(pk^0_{ID^*}, sk^0_{ID^*})$ and $(pk^1_{ID^*}, sk^1_{ID^*})$ are generated, where $(pk^0_{ID^*}, sk^0_{ID^*})$ are derived from mpk and msk respectively, while $(pk^1_{ID^*}, sk^1_{ID^*})$ are derived from a freshly generated master key pair. After a uniformly random bit b is chosen, the subprocedure $\mathcal{A}_2$ is executed on input mpk and $pk^b_{ID^*}$. $\mathcal{A}_2$ gets access to oracles WalSign, PK and getState, where getState returns the current state of the wallet scheme. $\mathcal{A}$ wins the game, if its subprocedure $\mathcal{A}_2$ returns a bit b', such that $b' = b$. We define the advantage of an adversary $\mathcal{A}$ as its winning probability in game WUNL over random guessing.

Definition 10 (Unlinkability). *Let* SW *be a stateful wallet scheme (cf. Definition 8). We say that* SW *is pq-unlinkable if for any quantum adversary $\mathcal{A}$, the advantage in game* WUNL *(cf. Figure 5) is negligible.*

3.2 Wallet Unforgeability

At a high level, unforgeability for stateful wallets ensures that funds held by the cold wallet remain secure even in case an adversary corrupts the hot wallet and/or observes transactions on the ledger signed by the cold wallet. In order to model this property, we define the game WUF, in which the adversary $\mathcal{A}$ obtains a master public key mpk and the initial state St_0 as input. This models the situation in which an adversary corrupts the hot wallet right after initialization of the wallet scheme. Further, $\mathcal{A}$ gets access to the oracles PK and WalSign, which are defined in the same way as in the game WUL, with the difference that WalSign now additionally keeps track of all queried messages. Eventually, $\mathcal{A}$ outputs a forgery consisting of a message m^*, a signature σ^* and an ID^*. $\mathcal{A}$ wins the game if (1) m^* has not been queried to WalSign before, (2) PK has been previously queried on ID^* and (3) σ^* is a valid signature for m^* under public key pk_{ID^*}.

```
Game WUF
$(mpk, msk, St) \leftarrow$ SW.KGen()
$b \leftarrow_\$ \{0, 1\}$
$(m^*, \sigma^*, ID^*) \leftarrow \mathcal{A}^{\mathsf{WalSign},\mathsf{PK}}(mpk, St)$
if $(Keys[ID^*] \neq \bot)$ then
    return 0
$(pk_{ID^*}, sk_{ID^*}) \leftarrow Keys[ID^*]$
if $(m^* \in M[ID^*])$ then
    return 0
if (SW.Verify$(pk_{ID^*}, m^*, \sigma^*) = 0$) then
    return 0
return 1
```

```
Oracle PK$(ID)$
$(pk_{ID}, St^*) \leftarrow$ SW.RandPK$(mpk, ID, St)$
$(sk_{ID}, St^*) \leftarrow$ SW.RandSK$(msk, ID, St)$
$Keys[ID] \leftarrow (pk_{ID}, sk_{ID})$
$St \leftarrow St^*$
return $pk_{ID}$
```

```
Oracle WalSign$(m, ID)$
if $(Keys[ID] = \bot)$ then
    return $\bot$
$(pk_{ID}, sk_{ID}) \leftarrow Keys[ID]$
$\sigma \leftarrow$ SW.Sign$(sk_{ID}, m)$
$M \leftarrow_\cup \{m\}$
return $\sigma$
```

Fig. 6. Unforgeability game WUF for stateful wallets.

Note that the adversary knows *mpk* and St_0 and hence can generate any session public key for any *ID* itself, which seems to make the PK oracle redundant. However, PK is still needed for bookkeeping purposes, i.e., to ensure that the session key pair for $\mathcal{A}$'s forgery has been created before $\mathcal{A}$ outputs its forgery. We define the advantage of an adversary $\mathcal{A}$ as its probability of winning the game WUF.

As mentioned in [16], the fact that the adversary can derive arbitrary session public keys makes the wallet scheme vulnerable to *related key attacks*, in case the underlying signature scheme is prone to such attack. Intuitively, upon an adversary learning a signature σ_{ID} and a corresponding session public key pk_{ID}, a related key attack allows the adversary to transform σ_{ID} into a valid signature σ_{ID^*} under public key pk_{ID^*}. This attack may have a severe impact on the security guarantees of our wallet scheme, since it may allow an adversary to steal all funds of a cold wallet. One common counter measure against related key attack used in [16, 35] is called *public key prefixing*, i.e., a signature on a message μ is computed as $\mathtt{Sign}(sk, (pk, \mu))$. In many signature schemes the signature is computed on the hash of the message and not the message itself. Therefore by prefixing the public key an adversary not only has to transform σ_{ID} into a valid signature σ_{ID^*} under public key pk_{ID^*} but also find a collision for the hash function in order to mount a related key attack.

Definition 11 (Unforgeability). *Let* SW *be a stateful wallet scheme (cf. Definition 8). We say that* SW *is pq-unforgeable if for any quantum adversary $\mathcal{A}$, the advantage in game* WUF *(cf. Figure 6) is negligible.*

In Appendix C we show why our relaxed notions of *signature schemes with rerandomizable public keys* (Definition 4) and EUF-CMA-HRK (Definition 6) are sufficient in the wallet setting.

3.3 Post-Quantum Security of Wallets

In this section we show that the generic construction achieves both unlinkability and unforgeability against quantum adversaries. Recall that since we are in the post-quantum setting, the oracles provided by the challenger in the unlinkability game (PK, getState, WalSign) and in the unforgeability game (PK, WalSign) are run on a classical computer. Hence, also the (quantum) adversary gets only classical access to these oracles. However, the adversary can use its quantum computing power to access the quantum random oracle $|\mathsf{H}\rangle$, i.e., querying the random oracle in superposition.

The following theorem shows the unlinkability.

Theorem 1. *Let* RSig *be a signature scheme with rerandomizable public keys (cf. Definition 4) and* H *a random oracle. Then the stateful wallet scheme* SW *built from* RSig *and* H *(cf. Figure 4) is pq-unlinkable according to Definition 10, i.e., against quantum adversaries which have access to* $|\mathsf{H}\rangle$.

Proof Sketch. Here, we provide a proof intuition of Theorem 1. For the full proof we refer to Appendix B. Let us first recall how the unlinkability property is proven in the classical ROM setting (cf. [16]). Note that the wallet public keys are derived from the wallet state, which is stored within the wallet, hidden from the adversary. The classical adversary can then try to guess one of the states of the wallet and make a "problematic query" to the random oracle H on such a state, in order to derive one of the session public keys generated by the wallet. If the adversary guesses the wallet's state correctly, it can distinguish a public key generated by the wallet from a randomly generated one, and hence the adversary will be able to win the unlinkability game. The classical proof consists of two steps: (1) showing the probability that the adversary makes the above mentioned *problematic* query to the random oracle is negligible, and (2) showing that the adversary has no advantage in winning the unlinkability game conditioned on the event that it does not make any problematic query. Finally, note that while this proof uses the stronger notion of rerandomizable public and secret keys (cf. Definition 3), it is easy to see that it also works with our relaxed definition of rerandomizable public keys (cf. Definition 4). This is because the unlinkability game requires the adversary to distinguish a public key generated by the wallet from a randomly generated public key.

Our proof in the QROM follows the same approach, however, the first step requires a different technique. Recall that the wallet state gets refreshed with every public key query. In [16], the challenger keeps a list of the states of the wallet scheme – starting from the initial state till the one obtained during last public key query. In the analysis a simple comparison allows to check whether a query by the classical adversary is problematic, i.e., whether it coincides with one of the states of the wallet. Since the adversary can access the random oracle H only classically (it can query on exactly one input at a time), hence the challenger can store all these queries in a list. In the QROM, however, we cannot keep such a list as the adversary now has quantum computation power, hence can query the random oracle on several, and even all, inputs in superposition.[5] Instead, we consider a game hop which we can bound by the advantage of the adversary in distinguishing two random oracles, which in turn can be bound using the O2H lemma. For the resulting game, we can show via a reduction to the rerandomizability property of public keys of the RSig scheme using the simulatability property of RSig that the adversary has only negligible advantage in winning the game.

The following theorem shows that the generic construction is unforgeable in the presence of quantum attackers.

Theorem 2. *Let* RSig *be a signature scheme with rerandomizable public keys (cf. Definition 4) and* H *a random oracle. Then the stateful wallet scheme* SW *built from* RSig *and* H *(cf. Figure 4) is pq-unforgeable according to Definition 11, i.e., against quantum adversaries which have access to* $|\mathsf{H}\rangle$.

We briefly recap the classical proof in the ROM (cf. [16]), thereby highlighting the challenge when switching to a quantum adversary. Note again, that the classical proof uses the stronger notion of rerandomizable keys (cf. Definition 3) but also holds for the weaker notion of rerandomizable public keys (cf. Definition 4). The proof consists of a game hop in which the adversary loses the game if there is a collision of session keys for different identities. Due to the construction, this occurs if the random oracle outputs a collision which is bound by a simple counting argument. The advantage of an adversary in the resulting game is bound by the security of the underlying signature scheme using a reduction. The crucial part is that the reduction simulates the random oracle H for the adversary using

[5] We note that, to some extent, the *compressed oracle technique* by Zhandry [47] allows the recording of superposition queries.

its oracle Rand from the EUF-CMA-HRK game. More precisely, for each query to H by the adversary, the reduction makes a query to Rand.

Our proof in the QROM follows the same idea, however additionally needs to take care of the use of the quantum random oracle $|\mathsf{H}\rangle$ by the adversary. The first part works exactly as in [16], since the access to the oracle PK remains classical even for a quantum adversary. The second part, however, does not work as in [16]. While the adversary can query the quantum random oracle $|\mathsf{H}\rangle$ in superposition, the reduction can query its oracle Rand only classical as it is provided by its (classical) challenger. By querying $|\mathsf{H}\rangle$ on an equal superposition of all (i.e., exponentially many) inputs, the reduction would need exponentially many queries to Rand in order to simulate $|\mathsf{H}\rangle$ for the adversary. Clearly, this would render the reduction useless as it would not be efficient. To tackle this issue, we do an additional game hop in which we switch from a random oracle to an oracle drawn from a small-range distribution. While this affects the advantage of the adversary only negligibly, it allows us to construct a reduction which can simulate the quantum random oracle for the adversary by making a polynomial number of (classical) queries to its oracle Rand.

Proof (Proof of Theorem 2.). Let $\mathcal{A}$ be an adversary which makes q_{H} queries to $|\mathsf{H}\rangle$. The proof consists of the following three games.

Game G_0: This game is the game WUF instantiated with SW (cf. Figure 4). Assume that $\mathcal{A}$ has non-negligible advantage $\epsilon = \epsilon(\lambda)$ in winning G_0. This means that there exists a polynomial $p = p(\lambda)$ such that $p(\lambda) > \frac{1}{\epsilon(\lambda)}$.

Game G_1: This game is the same as G_0, except the adversary loses when there is a collision of keys for different identities. To detect the change, the adversary has to make queries to PK which result in colliding keys. Note that the adversary only has classical access to PK as it is provided by the classical challenger. Hence the bound from [16] is applicable, which is a simple counting argument over the number of queries to PK. This yields that the advantage of $\mathcal{A}$ in G_1 is $\epsilon - \text{negl}(\lambda)$, i.e., it is negligibly close to its advantage ϵ in G_0.

Game G_2: In this game the adversaries queries to $|\mathsf{H}\rangle$ is simulated using Definition 1 and the Lemma 2. Let $l = 2Cq_{\mathsf{H}}^3 p$ with C being the constant from Lemma 2 and p being the polynomial described above. At the start of the game, the challenger will generate l random values and draw the first output (the randomness ρ) of the quantum random oracle $|\mathsf{H}\rangle$ from a small-range distribution using these l samples. The second output (the new state St) is generated just as in G_1. According to Lemma 2, $\mathcal{A}$ can only distinguish this game from the previous one with probability less than $\frac{1}{2p}$. Therefore, Lemma 2 yields that the advantage of $\mathcal{A}$ in this game is at least $\epsilon - \text{negl}(\lambda) - \frac{1}{2p}$.

Bounding the advantage of $\mathcal{A}$ in Game G_2: We now show how to transform an adversary $\mathcal{A}$ playing G_2 into an adversary $\mathcal{B}$ playing EUF-CMA-HRK (where, the underlying signature scheme is RSig). W.l.o.g., we assume that $\mathcal{A}$ never makes a query which results in $\perp$ and that there are no collisions. At the start, $\mathcal{B}$ receives a public key pk. It performs $l = 2Cq_{\mathsf{H}}^3 p$ queries to its oracle Rand and samples an initial state St_0. It invokes $\mathcal{A}$ on input $(mpk = pk, St_0)$.

Simulation of Quantum Random Oracle $|\mathsf{H}\rangle$*.* $\mathcal{B}$ simulates the first output (the randomness ρ), by using the l samples from Rand drawn from a small-range distribution. Note that Rand internally stores the output ρ in its list RList. For the second output (the new state St), $\mathcal{B}$ simulates it using a $2q_{\mathsf{H}}$-wise independent function which is indistinguishable for an adversary making q_{H} queries [46].

Simulation of PK *oracle.* When $\mathcal{A}$ queries its oracle PK on ID, $\mathcal{B}$ computes $pk_{ID} \leftarrow \mathtt{RSig.RandPK}(pk; \omega_{ID})$, where $(\omega_{ID}, St^*) \leftarrow \mathsf{H}(St, ID)$, sets $Keys[ID] \leftarrow (pk_{ID}, \omega_{ID})$, and sends pk_{ID} to $\mathcal{A}$.

Simulation of WalSign *oracle.* When $\mathcal{A}$ makes a query (m, ID) to its oracle WalSign, $\mathcal{B}$ obtains the $(pk_{ID}, \omega_{ID}) = Keys[ID]$, sets $m' \leftarrow (m, pk_{ID})$, queries its own oracle OHR on (m', ω_{ID}), and forwards the response to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery (m^*, σ^*, ID^*), $\mathcal{B}$ obtains $(pk_{ID^*}, \omega_{ID^*}) = Keys[ID^*]$, sets $\hat{m^*} \leftarrow (m^*, pk_{ID^*})$, and outputs $(\hat{m^*}, \sigma^*, \omega_{ID^*})$.

If $\mathcal{A}$'s forgery (m^*, σ^*, ID^*) is valid in Game G_2, then $\mathcal{B}$'s forgery $(\hat{m^*}, \sigma^*, \omega_{ID^*})$ is also valid in EUF-CMA-HRK. We now show that the output of $\mathcal{B}$ is a valid forgery whenever the output of $\mathcal{A}$ is. First, since (m^*, σ^*, ID^*) is a valid forgery by $\mathcal{A}$, we know that $\mathcal{A}$ never queried (m^*, ID^*) to $\mathsf{WalSign}$. Recall that, for every $\mathsf{WalSign}$ query by $\mathcal{A}$ on any message (m), $\mathcal{B}$ made a OHR query on public key prefixed message $(m' \leftarrow \{m, pk\})$. Since $\mathcal{A}$ never queried $\mathsf{WalSign}$ on input (m^*, ID^*), $\mathcal{B}$ never queried $(\hat{m^*}, \omega_{ID^*})$ to its oracle OHR, where $\hat{m^*} \leftarrow \{m^*, pk\}$. Second, it holds that $\omega_{ID^*} \in \mathsf{RList}$. This follows from the simulation of the quantum random oracle where, for every possible output (ρ, St), ρ is in RList. Third, validity of the forgery by $\mathcal{A}$ yields validity of the forgery by $\mathcal{B}$.

Recall that $l = 2Cq_{\mathsf{H}}^3 p$ and as discussed at the beginning of this game, according to Lemma 2, the advantage of the adversary in this game is equal to $\epsilon - \mathrm{negl}(\lambda) - \frac{1}{2p}$. Assuming the security of the underlying signature scheme $\mathtt{RSig}$, we have that this advantage must be negligible. Combined with $\epsilon > \frac{1}{p}$ (see description of G_0), this yields that $\frac{1}{2p}$ is negligible, resulting in a contradiction. Hence, we conclude that ϵ, the advantage of $\mathcal{A}$, is negligible.

4 PQ Signatures with Honestly Rerandomizable Public Keys

In this section we propose a lattice-based construction of a signature scheme with honestly rerandomizable public keys (cf. Definition 4). In such a signature scheme, the distribution of honestly rerandomized public keys is computationally indistinguishable to the distribution of original public key, while honestly rerandomized secret keys are allowed to be distributed differently from the original secret key. The scheme extends the generic construction of lattice-based signatures from Section 2.4. We analyze the security of our scheme in Section 4.2. In Appendix D we discuss alternative ways of key rerandomization in a lattice-based signature scheme and argue why they fall short in building practical hot/cold wallets.

4.1 Description of the Scheme

Let $\mathsf{LB}.\Sigma = (\mathsf{LB}.\mathtt{KGen}, \mathsf{LB}.\mathtt{Sign}, \mathsf{LB}.\mathtt{Verify})$ be the lattice-based signature scheme given in Section 2.4, Figure 3, and let $\mathbf{A} \in R_q^{k_1 \times k_2}$ be a uniformly random matrix as defined in Section 2.4, i.e., $\mathbf{A}$ is publicly known and an implicit input to all algorithms. Furthermore, we define the following functions and algorithms:

- Max_j is a function that on input $a \in R$, it outputs the j^{th} largest absolute coefficient of a. This function is used for bounding the secret-related terms, and hence the signatures generated by the algorithm $\mathsf{LB}.\mathtt{Sign}$ (cf. line 6–7 in Figure 3).
- GenG is an algorithm that on input $(\mathsf{dim}, \sigma, \mathsf{bnd}, \mathsf{rnd})$, it outputs a vector $\mathbf{x} = (x_1, \ldots, x_{\mathsf{dim}})$, where $x_i \in R$ are sampled from $D_{\mathbb{Z}^n,\sigma}$ such that $\sum_{j=1}^{\kappa} \mathsf{Max}_j(x_i) \leq \mathsf{bnd}$ by using a randomness rnd_i that is extracted from rnd, e.g., via the function E.
- $\mathsf{F} : \{0,1\}^* \longrightarrow \{0,1\}^{o(\lambda)}$ is a collision resistant hash function. It is used to hash the public key in order to prevent related key attacks [34].

In this section we set the distribution used in $\mathsf{LB}.\mathtt{KGen}$ for the secret key to $\chi = D_{\mathbb{Z}^n,\sigma}$. More precisely, we assume that $sk = (\mathbf{s}, \mathbf{e}) \in D_{\mathbb{Z}^n,\sigma}^{k_2} \times D_{\mathbb{Z}^n,\sigma}^{k_1}$, where $\mathbf{s} \leftarrow \mathsf{GenG}(k_2, \sigma, S/2, \mathsf{rnd}_s)$ and $\mathbf{e} \leftarrow \mathsf{GenG}(k_1, \sigma, E/2, \mathsf{rnd}_e)$ for two predefined positive numbers S, E and randomnesses $\mathsf{rnd}_s, \mathsf{rnd}_e$. Setting

```
RandG(ρ)
  1: $\rho := (\rho_s, \rho_e) \in \{0,1\}^{2o(\lambda)}$
  2: $\mathbf{r} \leftarrow \mathsf{GenG}(k_2, \sigma, S/2, \rho_s)$
  3: $\mathbf{u} \leftarrow \mathsf{GenG}(k_1, \sigma, E/2, \rho_e)$
  4: return $(\mathbf{r}, \mathbf{u})$

RSig.KGen(1^λ)
  1: $(sk, pk) \leftarrow \mathsf{LB}.\mathtt{KGen}(1^\lambda)$
  2: $\mathsf{hpk} \leftarrow \mathsf{F}(pk)$
  3: $sk \leftarrow (\mathsf{hpk}, sk)$
  4: return $(sk, pk)$

RSig.RandPK(pk, ρ)
  1: $(\mathbf{r}, \mathbf{u}) \leftarrow \mathsf{RandG}(\rho)$
  2: $\mathbf{b}' \leftarrow \mathbf{b} + \mathbf{A}\mathbf{r} + \mathbf{u} \pmod q$
  3: $pk' := \mathbf{b}'$
  4: return $pk'$

RSig.RandSK(sk, ρ)
  1: $(\mathbf{r}, \mathbf{u}) \in D^{k_2}_{\mathbb{Z}^n,\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sigma} \leftarrow \mathsf{RandG}(\rho)$
  2: $\mathbf{s}' \in D^{k_2}_{\mathbb{Z}^n,\sqrt{2}\sigma} \leftarrow \mathbf{s} + \mathbf{r}$
  3: $\mathbf{e}' \in D^{k_1}_{\mathbb{Z}^n,\sqrt{2}\sigma} \leftarrow \mathbf{e} + \mathbf{u}$
  4: $\mathbf{b}' \leftarrow \mathbf{A}\mathbf{s}' + \mathbf{e}' \pmod q$
  5: $\mathsf{hpk}' \leftarrow \mathsf{F}(\mathbf{b}')$
  6: $sk' := (\mathsf{hpk}', \mathbf{s}', \mathbf{e}')$
  7: return $sk'$

RSig.Sign(sk, m)
  1: $\mu \leftarrow (m, \mathsf{hpk})$
  2: $(\mathbf{z}_1, \mathbf{z}_2, c) \leftarrow \mathsf{LB}.\mathtt{Sign}(sk, \mu)$
  3: return $(\mathbf{z}_1, \mathbf{z}_2, c)$

RSig.Verify(pk, m, (z_1, z_2, c))
  1: $\mu \leftarrow (m, \mathsf{F}(pk))$
  2: return $\mathsf{LB}.\mathtt{Verify}(pk, \mu, (\mathbf{z}_1, \mathbf{z}_2, c))$
```

Fig. 7. Construction of lattice-based signature scheme with honestly rerandomizable public keys.

$\chi = D_{\mathbb{Z}^n,\sigma}$ is essential for rerandomizing the secret key in the construction introduced in this section because the sum of Gaussian distributed elements with standard deviation σ is also Gaussian distributed with standard deviation $\sqrt{2}\sigma$ (cf. Lemma 3).

In the following we describe our signature scheme with honestly rerandomizable public keys. The respective algorithms are formalized in Figure 7. In order to simplify the construction, we first define the algorithm RandG (see Figure 7 for a formal description). It takes as input a randomness $\rho = (\rho_s, \rho_e) \in \{0,1\}^{o(\lambda)} \times \{0,1\}^{o(\lambda)}$, and outputs two vectors $\mathbf{r}, \mathbf{u}$, which are generated by running the algorithm GenG on input $(k_2, \sigma, S/2, \rho_s)$, $(k_1, \sigma, E/2, \rho_e)$, respectively.

RSig.KGen: The key generation algorithm runs $\mathsf{LB}.\mathtt{KGen}$ to obtain key pair (sk, pk), where $sk = (\mathbf{s}, \mathbf{e}) \in D^{k_2}_{\mathbb{Z}^n,\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sigma}$ and $pk = \mathbf{b} \in R_q^{k_1}$. Then, it computes $\mathsf{hpk} = \mathsf{F}(pk)$, prepends hpk to sk, and returns the updated (sk, pk).

RSig.RandPK: Given public key $pk = \mathbf{b}$ and honestly generated randomness ρ, algorithm $\mathtt{RSig.RandPK}$ runs $\mathsf{RandG}(\rho)$ to generate a pair of Gaussian vectors $(\mathbf{r}, \mathbf{u})$. Then, it computes $\mathbf{b}' = \mathbf{b} + \mathbf{A}\mathbf{r} + \mathbf{u} \pmod q$ and outputs the honestly rerandomized public key $pk' = \mathbf{b}'$.

RSig.RandSK: Given $sk = (\mathsf{hpk}, \mathbf{s}, \mathbf{e})$ and honestly generated randomness $\rho \in \{0,1\}^{2o(\lambda)}$, the algorithm $\mathtt{RSig.RandSK}$ runs RandG to obtain $(\mathbf{r}, \mathbf{u}) \in D^{k_2}_{\mathbb{Z}^n,\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sigma}$. Then, it computes $\mathbf{s}' = \mathbf{s} + \mathbf{r}$ and $\mathbf{e}' = \mathbf{e} + \mathbf{u}$. Note that by Lemma 3, the pair $(\mathbf{s}', \mathbf{e}')$ is distributed as $D^{k_2}_{\mathbb{Z}^n,\sqrt{2}\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sqrt{2}\sigma}$. Finally, the algorithm computes $\mathsf{hpk}' = \mathsf{F}(\mathbf{b}')$ and outputs the honestly rerandomized secret key $sk' = (\mathsf{hpk}', \mathbf{s}', \mathbf{e}')$.

RSig.Sign: Algorithm $\mathtt{RSig.Sign}$ returns the signature obtained by calling $\mathsf{LB}.\mathtt{Sign}$ on message $\mu = (m, \mathsf{hpk})$. Signing messages together with the hash value of (honestly rerandomized) public keys ensures security under related key attacks [34].

RSig.Verify: Algorithm $\mathtt{RSig.Verify}$ returns the bit obtained by running $\mathsf{LB}.\mathtt{Verify}(pk, \mu)$, where $\mu = (m, \mathsf{F}(pk))$.

We note that rerandomizing sk must be carried out only with the original secret key, i.e., a rerandomized secret key cannot be used to generate a new rerandomized one. This ensures that all honestly rerandomized secret keys have identical distribution, i.e., $D^{k_2}_{\mathbb{Z}^n,\sqrt{2}\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sqrt{2}\sigma}$. Furthermore, signatures

```
Reduction $\mathcal{D}(pk)$
1: $\mathsf{RList} := \varnothing$
2: $\mathcal{Q} := \varnothing$
3: $(m, ((\mathbf{z}_1, \mathbf{z}_2, c), \rho)) \leftarrow \mathcal{A}^{\mathsf{H}', \mathsf{Rand}, \mathsf{OHR}}(pk)$
4: if $(\rho = \mathsf{NULL})$ then
5:     $\mathsf{hpk} \leftarrow \mathsf{F}(pk)$
6:     $\mu \leftarrow (m, \mathsf{hpk})$
7:     return $(\mu, (\mathbf{z}_1, \mathbf{z}_2, c))$
8: if $(\rho \neq \mathsf{NULL})$ then
9:     $pk' \leftarrow \mathtt{RSig.RandPK}(pk, \rho)$
10:    $\mathsf{hpk}' \leftarrow \mathsf{F}(pk')$
11:    $\mu' \leftarrow (m, \mathsf{hpk}')$
12:    $(\mathbf{r}, \mathbf{u}) \in D^{k_2}_{\mathbb{Z}^n,\sigma} \times D^{k_1}_{\mathbb{Z}^n,\sigma} \leftarrow \mathsf{RandG}(\rho)$
13:    $\mathbf{z}'_1 \leftarrow \mathbf{z}_1 - \mathbf{r}c$
14:    $\mathbf{z}'_2 \leftarrow \mathbf{z}_1 - \mathbf{u}c$
15: return $(\mu', (\mathbf{z}'_1, \mathbf{z}'_2, c))$
```

Fig. 8. Reduction from the EUF-CMA security of LB.Σ (Figure 3) to EUF-CMA-HRK security of signature scheme with honestly rerandomizable public keys (Figure 7). Queries to OHR, H', and Rand are answered as shown in Figure 9.

generated using honestly rerandomized keys have different distribution from signatures generated using the original key pair. More precisely, the pair $(\mathbf{z}_1, \mathbf{z}_2)$ is distributed uniformly at random over $R^{k_2}_{B_1} \times R^{k_1}_{B_2}$, where

$$B_1 = \begin{cases} Y - S/2 & \text{if } sk \leftarrow \mathsf{LB}.\mathtt{KGen} \\ Y - S & \text{if } sk \leftarrow \mathtt{RSig.RandSK} \end{cases}$$

$$B_2 = \begin{cases} Y - E/2 & \text{if } sk \leftarrow \mathsf{LB}.\mathtt{KGen} \\ Y - E & \text{if } sk \leftarrow \mathtt{RSig.RandSK} \end{cases}$$

The bound Y of the masking pair $(\mathbf{y}_1, \mathbf{y}_2)$ is chosen such that the probability of generating valid signatures (cf. Section 2.4) is at least $1/M$, i.e., $\left(\frac{2B_1+1}{2Y+1}\right)^{k_2 n} \cdot \left(\frac{2B_2+1}{2Y+1}\right)^{k_1 n} \geq 1/M$, where $M = O(1)$ is the repetition rate of the signing algorithm.

4.2 Security Analysis

In this section we analyze the EUF-CMA-HRK security of the scheme introduced in Section 4.1 in the QROM. More precisely, we reduce its EUF-CMA-HRK security to the EUF-CMA security of the lattice-based signature scheme $\mathsf{LB}.\Sigma = (\mathsf{LB}.\mathtt{KGen}, \mathsf{LB}.\mathtt{Sign}, \mathsf{LB}.\mathtt{Verify})$ described in Section 2.4. The correctness of the scheme directly follows from the correctness of LB.Σ. Note that rerandomizability of public keys (see Definition 4) follows from the MLWE assumption [28]. That is, for any public key $\mathbf{b}$ and any honestly rerandomized public key $\mathbf{b}'$ both pairs $(\mathbf{A}, \mathbf{b})$, $(\mathbf{A}, \mathbf{b}')$ are computationally indistinguishable from the uniform distribution over $R_q^{k_1 \times k_2} \times R_q^{k_1}$.[6] In addition, it is easy to show that the simulatability property (see Definition 4) is satisfied in the quantum random oracle model.

[6] We note that our scheme could even be instantiated such that it achieves statistical indistinguishability of public keys by sampling the secret key from a Gaussian distribution with somewhat larger standard deviation. However, in this work we focus on computational indistinguishability as it suffices in the wallet setting.

Theorem 3 (EUF-CMA-HRK Security). *The signature scheme with honestly rerandomizable public keys depicted in Figure 7 is EUF-CMA-HRK secure in the* QROM *if scheme* $LB.\Sigma = (LB.\mathtt{KGen}, LB.\mathtt{Sign}, LB.\mathtt{Verify})$ *described in Figure 3 is EUF-CMA secure in the* QROM.

Proof. Let $\mathcal{A}$ be an adversary that is able to generate valid forgeries under the signature scheme with honestly rerandomizable public keys, i.e., $\mathcal{A}$ is able to win the game $\mathsf{EUF\text{-}CMA\text{-}HRK}^{\mathcal{A}}_{\mathtt{RSig}}$ (cf. Definition 6). We construct an algorithm D that runs $\mathcal{A}$ as subroutine in order to win the game $\mathsf{EUF\text{-}CMA}^{\mathcal{D}}_{\mathsf{LB}.\Sigma}$ (see Definition 5) against the scheme $\mathsf{LB}.\Sigma$. According to the security model, $\mathcal{A}$ has quantum access to a random oracle H' and classical access to a random oracle Rand in addition to classical access to the signing oracle OHR. The reduction D has quantum access to the random oracle H and classical access to the signing oracle O, which returns to D signatures generated by $\mathsf{LB}.\Sigma$. The algorithm D is described in Figure 8. D initializes two empty lists $\mathsf{RList}, \mathcal{Q}$. These are used by D to store queries to Rand and OHR, respectively. Simulation of $\mathsf{OHR}, \mathsf{H}'$, and Rand is given in Figure 9.

Analysis. Let $(m, ((\mathbf{z}_1, \mathbf{z}_2, c), \rho))$ be a valid forgery output by $\mathcal{A}$. This means that $m \notin \mathcal{Q}$ and $\mathtt{RSig.Verify}(pk, m, (\mathbf{z}_1, \mathbf{z}_2, c)) = 1$. Moreover, $\rho \in \mathsf{RList}$ in case randomness $\rho \neq \mathsf{NULL}$.

We first analyze the case that $\rho = \mathsf{NULL}$. The signature satisfies $(\mathbf{z}_1, \mathbf{z}_2) \in R^{k_2}_{Y-\frac{S}{2}} \times R^{k_1}_{Y-\frac{E}{2}}$ and $c = \mathsf{H}'(\mathbf{w}, m, \mathsf{hpk}) = \mathsf{H}(\mathbf{w}, m, \mathsf{hpk})$, where $\mathbf{w} = \mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{b}c \pmod q$. Hence, this forgery constitutes a valid signature under $\mathsf{LB}.\Sigma$ on message $\mu = (m, \mathsf{hpk})$. Note that if c was not queried by some input, then $\mathcal{A}$ produces such c only with negligible probability, i.e., $1/|\mathbb{T}^n_\kappa|$. Thus, with probability of $1 - 1/|\mathbb{T}^n_\kappa|$, the value c must be a random oracle answer to a query made by $\mathcal{A}$, where $|\mathbb{T}^n_\kappa| = 2^\kappa \binom{n}{\kappa}$ and κ is chosen such that $|\mathbb{T}^n_\kappa| \geq 2^{2\lambda}$. This ensures that the probability of mapping two different values to the same output of H is at most $2^{-2\lambda}$.

Next, we assume that $\mathcal{A}$ outputs a valid forgery $(m, (\mathbf{z}_1, \mathbf{z}_2, c), \rho)$ under honestly rerandomized public key $\mathbf{b}'$ and $\rho \neq \mathsf{NULL}$. This means that $(\mathbf{z}_1, \mathbf{z}_2) \in R^{k_2}_{Y-S} \times R^{k_1}_{Y-E}$. In this case D transforms this signature into a forgery under the original public key $\mathbf{b}$ as follows: D runs $\mathsf{RandG}(\rho)$ to obtain $(\mathbf{r}, \mathbf{u})$. Then, it computes the vectors $\mathbf{z}'_1 = \mathbf{z}_1 - \mathbf{r}c$ and $\mathbf{z}'_2 = \mathbf{z}_2 - \mathbf{u}c$. Note that

$$\begin{aligned} \|\mathbf{z}'_1\|_\infty &\leq \|\mathbf{z}_1\|_\infty + \|\mathbf{r}c\|_\infty \leq \ Y - S + S/2 = Y - S/2, \\ \|\mathbf{z}'_2\|_\infty &\leq \|\mathbf{z}_2\|_\infty + \|\mathbf{u}c\|_\infty \leq Y - E + E/2 = Y - E/2 \ . \end{aligned}$$

Hence, $(\mathbf{z}'_1, \mathbf{z}'_2) \in R^{k_2}_{Y-\frac{S}{2}} \times R^{k_1}_{Y-\frac{E}{2}}$. Furthermore, we have

$$\mathbf{w} = \mathbf{A}\mathbf{z}'_1 + \mathbf{z}'_2 - \mathbf{b}c = \mathbf{A}(\mathbf{z}_1 - \mathbf{r}c) + \mathbf{z}_2 - \mathbf{u}c - \mathbf{b}c = \mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{b}'c \pmod q \ .$$

Therefore, it holds that $c = \mathsf{H}'(\mathbf{w}, m, \mathsf{hpk}') = \mathsf{H}(\mathbf{w}, m, \mathsf{hpk}')$. Hence, the forgery output by $\mathcal{A}$ can be turned into a valid forgery under the original public key $\mathbf{b}$ for message $\mu' = \mathsf{F}(m, \mathsf{hpk}')$, i.e., it is a forgery under $\mathsf{LB}.\Sigma$.

Finally, we note that the environment of $\mathcal{A}$ is perfectly simulated, and whenever $\mathcal{A}$ wins the game $\mathsf{EUF\text{-}CMA\text{-}HRK}^{\mathcal{A}}_{\mathtt{RSig}}$, D wins the game $\mathsf{EUF\text{-}CMA}^{\mathcal{D}}_{\mathsf{LB}.\Sigma}$. The number of signing queries made by D is at most $M \cdot Q$, where $M = O(1)$ is the repetition rate[7] of $\mathsf{LB}.\Sigma$ and Q is the number of signing queries made by $\mathcal{A}$.

[7] In practice, the repetition rate of the signing algorithm of standard lattice-based signature schemes is strictly smaller than 4 (e.g., see [7, 19]).

```
Sim($pk, m, \rho$)
1: if ($\rho = \mathsf{NULL}$) then
2:     $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3:     return SimNoR($pk, m$)
4: if ($\rho \neq \mathsf{NULL}$) then
5:     if ($\rho \notin \mathsf{RList}$) then
6:         return $\perp$
7:     $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
8:     return SimR($pk, m, \rho$)
```

```
SimNoR($pk, m$)
1: $\mathsf{hpk} \leftarrow \mathsf{F}(\mathbf{b})$
2: $\mu \leftarrow (m, \mathsf{hpk})$
3: $(\mathbf{z}_1, \mathbf{z}_2, c) \leftarrow \mathsf{O}(\mu)$
4: return $(\mathbf{z}_1, \mathbf{z}_2, c)$
```

```
H′(·)
1: return $\mathsf{H}(\cdot)$
```

```
SimR($pk, m, \rho$)
1: $(\mathbf{r}, \mathbf{u}) \leftarrow \mathsf{RandG}(\rho)$
2: $pk' := \mathbf{b}' \leftarrow \mathtt{RSig.RandPK}(pk, \rho)$
3: $\mathsf{hpk}' \leftarrow \mathsf{F}(\mathbf{b}')$
4: $\mu' \leftarrow (m, \mathsf{hpk}')$
5: $(\mathbf{z}_1', \mathbf{z}_2', c) \leftarrow \mathsf{O}(\mu')$
6: $\mathbf{z}_1 \leftarrow \mathbf{z}_1' + \mathbf{r}c$
7: $\mathbf{z}_2 \leftarrow \mathbf{z}_2' + \mathbf{u}c$
8: if $\big((\mathbf{z}_1, \mathbf{z}_2) \notin R^{k_2}_{Y-S} \times R^{k_1}_{Y-E}\big)$ then
9:     goto 5
10: return $(\mathbf{z}_1, \mathbf{z}_2, c)$
```

```
Rand()
1: $\rho \leftarrow_{\$} \{0,1\}^{2o(\lambda)}$
2: $\mathsf{RList} \leftarrow \mathsf{RList} \cup \{\rho\}$
3: return $\rho$
```

Fig. 9. Description of algorithm Sim, which simulates signing queries to OHR. The algorithms SimNoR, SimR are subroutines used by Sim. The first one is called when signing query does not include randomness ρ, while the latter one is called when signing query includes honestly generated randomness $\rho \neq \mathsf{NULL}$. Queries to H′ made by adversary $\mathcal{A}$ are redirected to the random oracle H, to which reduction D has access. Queries to Rand are answered locally by D.

5 Practical Instantiation

In this section we present an efficiency analysis of the wallet scheme introduced in Section 3. To this end, we instantiate the signature scheme presented in Section 4 with a concrete lattice-based signature scheme. The most recent Fiat-Shamir constructions of lattice-based signatures are Dilithium [19] and qTESLA [7]. We consider the latter scheme, since the hard lattice problem underlying its key generation algorithm uses Gaussian distributed secrets. This is essential for rerandomizing the secret key in our setting, and hence is sufficient for our scheme with honestly rerandomizable public keys described in Figure 7. On the other hand, Dilithium's key generation uses uniformly distributed secrets for the underlying lattice problem, instead of Gaussian distributed secrets, which is not suitable in our wallet setting (see Appendix D). Employing the Gaussian distribution in the key generation algorithm of Dilithium instead, requires to adjust the security analysis of Dilithium and to choose new parameters. The resulting scheme would be similar to qTESLA, with slight differences in how signatures are compressed. We choose not to modify Dilithium's original design but stick to qTESLA, which does not need any modification for our setting and is well-studied in comparison to a modified version of Dilithium.

In Appendix E we describe the technical details of instantiating our signature scheme with honestly rerandomizable public keys with qTESLA and show how its EUF-CMA-HRK security holds.

5.1 Deploying PQ Wallets over Blockchains

In this section we give an overview of the transaction throughput that can be achieved in a cryptocurrency system using our signature scheme instantiated with qTESLA.

A simple transaction in most cryptocurrency networks transfers coins from one party to another. Such transactions must usually include the public key pk and the signature σ of the sender such that the validity of the transaction can be verified. In order to give an estimated transaction throughput, we use the raw transaction size of a regular Bitcoin transaction (i.e., without the size of pk and σ) and then add the size of pk and σ of our scheme to it. The raw transaction size of a Bitcoin is roughly 100 Bytes (B) [5]. Hence, when instantiating our wallet scheme with qTESLA, we can take the corresponding signature size (2,592 B) and public key size (14,880 B) [7, Table 4] for a post-quantum security level of 95 bits[8] and add those to the rough raw transaction size of 100 B. The size of a transaction would then result in $100\text{ B} + 14,880\text{ B} + 2,592\text{ B} \approx 17.5$ KB. We note that it is possible for a party to send coins to multiple receivers in a single transaction which would essentially allow for transactions to be aggregated and increase efficiency.

Many cryptocurrencies (including Bitcoin and Ethereum) currently use the classical signature scheme ECDSA. For the sake of drawing a comparison, note that the size of the ECDSA public key and signature in Bitcoin is approximately 65 B and 73 B [4], respectively, which results in more compact transactions (minimum size of a transaction being $100B + 65B + 73B \approx 240B$), and hence higher transaction throughput.

Naturally, there are various ways to improve the transaction throughput such as increasing block size and the rate at which blocks are produced. For example, in a Bitcoin-like currency new blocks are created roughly every 10 minutes, which tremendously limits the throughput and scalability of the network. In contrast, one can consider a system with a block rate of a few seconds, say 15-20 seconds (e.g., this is the case for the Ethereum blockchain). This significantly increases transaction throughput, and hence compensates for larger sizes of pk and σ. Yet these solutions are ad-hoc, while a more interesting direction for future work is to design further efficient post-quantum secure signature schemes with rerandomizable keys.

Acknowledgments

The authors are grateful to the anonymous reviewers of ACM CCS'20 for their valuable comments. This work is supported by the German Research Foundation (DFG) Emmy Noether Program FA 1320/1-1, by the German Research Foundation DFG - SFB 1119 - 236615297 (CROSSING Projects P1 and S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain Project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE*.

References

1. Bitcoin post-quantum. https://bitcoinpq.org/. 2, 25
2. Quantum resistant ledger (qrl). https://github.com/theQRL/Whitepaper/blob/master/QRL_whitepaper.pdf. 2, 25
3. Bitcoin bip32 specification. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, Feb. 24 2017. Accessed: 2020-09-15. 2, 5
4. Bitcoin wiki: Elliptic curve digital signature algorithm, Nov. 2019. https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm. 22
5. Bitcoin wiki transaction format. https://en.bitcoin.it/wiki/Transaction, Dec. 2019. Accessed: 2020-05-04. 22
6. Divesh Aggarwal, Gavin Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. *Ledger*, 3, 2018. 2

[8] Note that qTESLA proposes only two parameter sets, chosen with respect to a conservative cost model: qTESLA-p-I with 95 bits of post-quantum security and qTESLA-p-III with 160 bits of post-quantum security [7, Section 4.3].

7. Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Krämer, Patrick Longa, and Jefferson E. Ricardini. The lattice-based digital signature scheme qTESLA. In *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020*, Lecture Notes in Computer Science, 2020. 4, 5, 11, 20, 21, 22, 27, 28
8. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, 2019*, volume 11693 of *Lecture Notes in Computer Science*, pages 269–295. Springer, 2019. 6
9. Shi Bai and Steven D Galbraith. An improved compression technique for signatures based on learning with errors. In *Cryptographers' Track at the RSA Conference*, pages 28–47. Springer, 2014. 5, 11, 27
10. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, 1993*, pages 62–73. ACM, 1993. 6
11. BitcoinExchangeGuide. CipherTrace Releases Report Exposing Close to $1 Billion Stolen in Crypto Hacks During 2018. https://bitcoinexchangeguide.com/ciphertrace-releases-report-exposing-close-to-1-billion-stolen-in_-crypto-hacks-during-2018/, 2018. 2
12. Bloomberg. How to Steal $500 Million in Cryptocurrency. http://fortune.com/2018/01/31/coincheck-hack-how/, 2018. 2
13. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7073, pages 41–69. Springer, 2011. 6
14. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography - PKC 2011*, pages 1–16. Springer, 2011. 10
15. Alexandru Cojocaru, Juan A. Garay, Aggelos Kiayias, Fang Song, and Petros Wallden. The bitcoin backbone protocol against quantum adversaries. Cryptology ePrint Archive, Report 2019/1150, 2019. https://eprint.iacr.org/2019/1150. 2
16. Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In *ACM SIGSAC Conference on Computer and Communications Security - CCS 2019*, pages 651–668. ACM, 2019. 1, 2, 3, 5, 9, 11, 12, 14, 15, 16, 26
17. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference*, volume 11693, pages 356–383. Springer, 2019. 5, 11
18. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *Advances in Cryptology–CRYPTO 2013*, pages 40–56. Springer, 2013. 5
19. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *Transactions on Cryptographic Hardware and Embedded Systems - TCHES 2018*, (1):238–268, 2018. 5, 11, 20, 21, 27
20. Muhammed F Esgin, Raymond K Zhao, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 567–584, 2019. 1, 2, 24
21. ethereum.org. Ethereum. https://ethereum.org/, 2015. 1
22. Chun-I Fan, Yi-Fan Tseng, Hui-Po Su, Ruei-Hau Hsu, and Hiroaki Kikuchi. Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. *International Journal of Information Security*, pages 1–11, 2019. 1, 5
23. Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *Public Key Cryptography - PKC 2016*, pages 301–330. Springer, 2016. 3, 4, 5, 7, 9, 26, 27
24. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science - ICS 2010*, pages 230–240. Tsinghua University Press, 2010. 26
25. Gus Gutoski and Douglas Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015*, volume 8975, pages 497–504. Springer, 2015. 1, 5
26. Timo Hanke, Mahnush Movahedi, and Dominic Williams. DFINITY technology overview series, consensus system. *CoRR*, abs/1805.04548, 2018. 1

27. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *Advances in Cryptology–EUROCRYPT 2018*, pages 552–586. Springer, 2018. 5, 11
28. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. 10, 19
29. Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019. 5, 11
30. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology–ASIACRYPT 2009*, pages 598–616. Springer, 2009. 5
31. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010*, pages 1–23. Springer, 2010. 11
32. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012*, pages 700–718. Springer, 2012. 10
33. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *Symposium on Foundations of Computer Science (FOCS 2004)*, pages 372–381. IEEE Computer Society, 2004. 10
34. Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the schnorr signature scheme and DSA against related-key attacks. In *Information Security and Cryptology - ICISC 2015*, volume 9558, pages 20–35. Springer, 2015. 17, 18
35. Hiraku Morita, Jacob CN Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the schnorr signature scheme and dsa against related-key attacks. In *ICISC 2015*, pages 20–35. Springer, 2015. 14
36. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. `http://bitcoin.org/bitcoin.pdf`. 1
37. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011. 6
38. Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. `http://eprint.iacr.org/2015/1098`. 1, 25
39. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. 3
40. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994. 2
41. Rhys Skellern. Cryptocurrency Hacks: More Than $2b USD lost between 2011-2018. `https://medium.com/ecomi/cryptocurrency-hacks-more-than-2b-usd-lost-between-2011-2018_-67054b342219`, 2018. 2
42. Mathieu Turuani, Thomas Voegtlin, and Michael Rusinowitch. Automated verification of electrum wallet. In *International Conference on Financial Cryptography and Data Security*, pages 27–42. Springer, 2016. 1, 5
43. Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. 6
44. Dominique Unruh. Post-quantum security of Fiat-Shamir. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security*, volume 10624, pages 65–95. Springer, 2017. 5, 11
45. Mark Zhandry. How to construct quantum random functions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 679–687. IEEE Computer Society, 2012. 7
46. Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference*, volume 7417, pages 758–775. Springer, 2012. 16
47. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference*, volume 11693, pages 239–268. Springer, 2019. 15

A Other Related Work

As mentioned in the introduction, various works build blockchains with security features against quantum adversaries. Most recently, Esgin et al. [20] have proposed a new ring signature scheme based on lattice assumptions for the blochchain setting which focus on similar anonymity guarantees to Monero [38]. In Monero-like cryptocurrencies the sender of a transaction can hide her identity in a set of transactions using ring signatures. In particular, the public key related to the sender's signature is never revealed explicitly in the blockchain network, hence remains unlinkable to the sender. We note that this notion of unlinkability is different from our notion of session key unlinkability.

Blockchain initiatives such as the "Bitcoin Post-Quantum" [1] and QRL [2] replace ECDSA with hash-based signature schemes which are post-quantum secure. Despite the hash-based schemes being quite efficient, the underlying hash function does not permit to construct a signature scheme with rerandomizable keys which plays a key role in our wallet scheme.

B Proof of Theorem 1

Proof. Throughout, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary which makes q and q_{PK} queries to its oracles $|\mathsf{H}\rangle$ and PK, respectively. To prove the theorem we use the following two games.

Game G_0: This game is the game WUNL instantiated with $\mathtt{SW}$ (cf. Figure 4).

Game G_1: This game is the same as G_0, except that the randomness ρ and the new state St^*, prior to running $\mathcal{A}_2$ (i.e., Line 13 in Figure 5), are sampled at random, independent of the random oracle. In both games, the randomness and new state are distributed identical. The only difference lies in the random oracle. From the point of view of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the random oracle in game G_1 is $|\mathsf{H}_{\mathcal{S}\to\$}\rangle$, i.e., the random oracle that is reprogrammed to random values for every $x \in \mathcal{S}$, where $\mathcal{S}$ contains all pairs of states and IDs prior to running $\mathcal{A}_2$. Hence, we can bound the advantage in distinguishing G_0 and G_1 by the advantage in distinguishing the random oracles $|\mathsf{H}\rangle$ and $|\mathsf{H}_{\mathcal{S}\to\$}\rangle$. Applying the O2H Lemma (cf. Lemma 1) yields

$$\left|\Pr\left[\mathcal{A}^{|\mathsf{H}\rangle} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{|\mathsf{H}_{\mathcal{S}\to\$}\rangle} \Rightarrow 1\right]\right| \leq 2q\sqrt{\Pr[x \in \mathcal{S} \colon \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow x]}\,.$$

where $\mathcal{B}$ is the adversary specified in Lemma 1. Note that $\mathcal{A}$ has no information about the states in the set $\mathcal{S}$ until it queries $\mathsf{getState}$ to which only $\mathcal{A}_2$ has access. Furthermore, we have $|\mathcal{S}| \leq q_{\mathsf{PK}} + 2$ at any point in time, q_{PK} from $\mathcal{A}_1$'s queries and 2 from the challenge phase. This yields

$$\Pr\left[x \in \mathcal{S} \colon \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow x\right] \leq \frac{|\mathcal{S}|}{2^\lambda} \leq \frac{q_{\mathsf{PK}} + 2}{2^\lambda}.$$

Combining the above equations, yields that the advantage in distinguishing G_0 and G_1 is negligible in λ.

It remains to bound the advantage of $\mathcal{A}$ in game G_1, where the same argument from the classical proof applies. In G_1, the challenge public key $pk^b_{ID^*}$ given to $\mathcal{A}_2$ is independent of the random oracle (as the random oracle is not used in G_1 anymore for deriving keys). Hence, it is irrelevant whether the adversary makes any query (classical or quantum) to the random oracle. We can show via reduction to the rerandomizability of public keys property of the $\mathtt{RSig}$ signature scheme that the challenge public keys $pk^0_{ID^*}$ and $pk^1_{ID^*}$ are computationally indistinguishable using the simulatability property of $\mathtt{RSig}$. This yields that the adversarial advantage is negligible. Combining the above proves the theorem.

C Relevance of Our Relaxed Notions

In Section 2 we defined the notions of *signature schemes with rerandomizable public keys* (cf. Definition 4) and $\mathsf{EUF\text{-}CMA\text{-}HRK}$ (cf. Definition 6). While these notions deviate from the ones used in previous works [16, 23], it turns out that our relaxed notions are sufficient for building deterministic wallets as we discuss below.

Rerandomizable public keys. One of the benefits of using deterministic wallets is that individual payments to the wallet are unlinkable (cf. Figure 5). To satisfy the unlinkability definition, Das et al. [16] require that the underlying signature scheme must have rerandomizable secret and public keys. However, as it can be observed in the unlinkability game, the adversary gets access only to the public key, while the secret key is never revealed to the adversary (as revealing it would trivially break the security of the scheme). Hence, it is sufficient to use our relaxed notion of rerandomizable public keys in order to achieve the unlinkability property. While the post-quantum secure signature scheme that we consider in this work does not offer rerandomizable public and secret keys as required by [16], it fortunately achieves our relaxed notion of rerandomizable public keys. Thus, it is sufficient to instantiate a wallet scheme that achieves the unlinkability property.

EUF-CMA-HRK. As in [16], we use the notion of *EUF-CMA under honestly rerandomizable keys*, where unforgeability holds if the randomness used to derive the keys is *honestly* generated. This is in contrast to the stronger notion of EUF-CMA-RK as defined in [23], where unforgeability must hold for *adversarial* chosen randomness. Stateful deterministic wallet schemes, however, derive the randomness deterministically from the state (see Figure 4), which is generated initially during a trusted setup when the master keys are created. Hence, the adversary has no influence on the randomness used during the rerandomization procedures. To conclude, the notion of EUF-CMA-HRK is not only suitable but also sufficient in the wallet setting.

D Alternative Methods for Rerandomization

In this section we describe alternative approaches for rerandomizing keys in the lattice setting and show why our scheme introduced in Section 4.1 is the most suitable option in the context of hot/cold wallets. First, we recall that our construction from the previous section assumes that the distribution of the secrets used in the key generation algorithm are from the Gaussian distribution, i.e., $\chi = D_{\mathbb{Z}^n,\sigma}$. This allows us to use Lemma 3 in order to obtain rerandomized secret keys that are also Gaussian distributed but with a slightly different standard deviation, i.e., $D_{\mathbb{Z}^n,\sqrt{2}\sigma}$. The key generation of our scheme cannot use uniformly distributed secrets over a small subset R_d from R, where R_d is the set of all polynomials from R with ℓ_∞ norm bounded by some integer $d \geq 1$. This is because the sum of two uniformly random polynomials over R_d does not yield a polynomial that follows the uniform distribution over a subset $S \subseteq R_d$. Using a uniformly random sk for rerandomization would yield rerandomized secret keys with unknown distribution, and hence the hardness of the computational assumption underlying the rerandomized key pairs would be unclear. Let us now discuss the alternative approaches.

Rerandomizability of Gaussian distributed secret keys. It is (theoretically) possible to rerandomize key pairs such that the rerandomized secret keys have the same distribution as the original secret key. More precisely, assume that sk is Gaussian distributed with standard deviation σ. Given a randomness ρ, a rerandomized secret key is computed as $sk' = sk + \rho$. Due to [24, Lemma 3], sk' is Gaussian distributed with the same σ when σ is of a super-polynomial size in the security parameter λ. In other words, we must select σ large enough in order to make the statistical distance between the distribution of sk and sk' negligible in λ. This value of σ gives secret keys of very large size, and requires to increase the size of the masking vectors used in the signing algorithm. Hence, we obtain signatures of very large size, which rules out using the resulting scheme in practice.

Rerandomizability of uniform distributed secret keys. In theory, it is possible to use uniformly distributed rather than Gaussian distributed secret keys as follows. Assume that $\chi = R_d$ and $\rho \in R_1$. The rerandomized secret key $sk' = sk + \rho$ is uniformly distributed over R_{d-1} with probability

$\left(\frac{2d-1}{2d+1}\right)^{(k_1+k_2)n}$, where $(k_1+k_2)n$ is the dimension of sk'. Therefore, for a very large d this probability would be overwhelming in λ.

Example 1. By considering the parameters of Dilithium [19] proposed for $\lambda = 128$, we have $k_1 = 5$, $k_2 = 4$, and $n = 256$. Hence, we have to set $d \approx 2^{139}$ in order to make the previously stated probability at least $1 - 2^{-128}$. This value of d yields a secret key of size $\approx 2^{147}$ Bytes.

The above given example shows that this approach is merely of theoretical interest only and is not suitable for practical applications as it requires huge sizes of keys, and hence signatures.

Allowing rerandomization algorithms to communicate. Consider an application, in which the rerandomization algorithms (i.e., `RSig.RandSK` and `RSig.RandPK`) synchronize after each invocation of `RSig.RandSK`. Given sk and ρ, the algorithm `RSig.RandSK` uses ρ together with a counter ctr in order to deterministically generate a randomness ρ', e.g., by using the function E on input (ρ, ctr). Then, it computes $sk' = sk + \rho'$ and outputs the rerandomized secret key sk' only after verifying that it has the correct distribution. If this is not the case, it increases ctr by 1 and repeats this process. The algorithm `RSig.RandPK` needs to receive the corresponding ctr from `RSig.RandPK` in order to generate the rerandomized public key related to sk'. Note that if sk is Gaussian distributed, then we even obtain a scheme with rerandomizable public and secret keys as defined in [23]. While this method is practical and may be applicable in the construction of sanitizable signatures proposed in [23], it cannot be used in the setting of hot/cold wallets due to the fact that in each signing process `RSig.RandPK` must obtain the correct ctr that were used to generate sk'. This synchronization requirement undermines the main concept of hot/cold wallets, namely the fact that hot and cold wallets do not communicate with each other (except when they are being initialized).

E An Instantiation with qTESLA

In this section we show how the signature scheme with honestly rerandomizable public keys introduced in Section 4 can be instantiated with qTESLA. We note that the parameters of qTESLA were selected according to the security reduction from the RLWE problem. This approach has two different aspects: On the one hand, it guarantees that qTESLA has the security level as long as the underlying RLWE instance is hard enough. On the other hand, this approach affects the performance and sizes of keys and signatures, because larger parameters are required to achieve the desired security level. The main goal of our choice is to demonstrate that our wallet scheme can be instantiated with state-of-the-art lattice-based signature schemes without taking into account any of the two different aspects mentioned above.

The design of our scheme is based on lattices over modules. In order to employ qTESLA in our construction we set $k_2 = 1$ to obtain a variant based on lattices over ideals, and security based on the hardness of RLWE. The (master) secret key includes polynomials $s, e_1, \ldots, e_{k_1}$ sampled from $D_{\mathbb{Z}^n,\sigma}$. The polynomial s is bounded by $S/2$ using the function Max_j defined in Section 2.4, while $e_1, \ldots, e_{k_1}$ are each bounded by $E/2$ using Max_j. In qTESLA the bounds are S and E, respectively. However, our wallet scheme uses the master key pair only for rerandomization, and signatures are generated using honestly rerandomized key pairs, which already satisfy the bounds S and E. Therefore, we can use exactly the same parameters proposed for qTESLA in [7, Table 4]).

Note that in comparison to the generic signature scheme shown in Figure 3, Section 2.4, the signature scheme qTESLA [7] compresses signatures by employing the technique of [9]. In this technique the signer proves knowledge of only the secret polynomial s rather than s and $e_1, \ldots, e_{k_1}$. Therefore, signatures are of the form $(z_1, c) \in R_Y \times \mathbb{T}^n_\kappa$ rather than $(z_1, \mathbf{z}_2, c) \in R_Y \times R_Y^{k_1} \times \mathbb{T}^n_\kappa$. This approach does not affect the EUF-CMA-HRK security of the signature scheme with honestly rerandomizable public keys. That is, the reduction given in Figure 8 remains the same. Only simulating the signing oracle (cf. Figure 9) requires to include an additional check to ensure the correctness of simulated signatures. More concretely, after step 9 of algorithm SimR (see Figure 9) we add the last **for** loop of qTESLA's signature generation algorithm [7, Algorithm 4]. However, we have in our setting

$$w_i = a_i z_1 - b'_i c - r_i c \ (\mathsf{mod}^{\pm} q) \ \text{ for all } \ i = 1, \ldots, k_1,$$

where a_i, b'_i, and r_i are the entries of the public vector $\mathbf{a}$ (replaced by the matrix $\mathbf{A}$, since $k_2 = 1$), rerandomized public key $\mathbf{b}'$, and the vector $\mathbf{r}$, respectively.

D. Deterministic Wallets for Adaptor Signatures

In this chapter, we present the following publication with minor changes:

[83] A. Erwig and S. Riahi. "Deterministic Wallets for Adaptor Signatures". In: *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II.* 2022, pp. 487–506. **Part of this thesis.**

Deterministic Wallets for Adaptor Signatures

Andreas Erwig(✉) and Siavash Riahi

Technische Universität Darmstadt, Germany
`firstname.lastname@tu-darmstadt.de`

Abstract. *Adaptor signatures* are a new cryptographic primitive that binds the authentication of a message to the revelation of a secret value. In recent years, this primitive has gained increasing popularity both in academia and practice due to its versatile use-cases in different Blockchain applications such as atomic swaps and payment channels. The security of these applications, however, crucially relies on users storing and maintaining the secret values used by adaptor signatures in a secure way. For standard digital signature schemes, cryptographic wallets have been introduced to guarantee secure storage of keys and execution of the signing procedure. However, no prior work has considered cryptographic wallets for adaptor signatures.
In this work, we introduce the notion of *adaptor wallets*. Adaptor wallets allow parties to securely use and maintain adaptor signatures in the Blockchain setting. Our adaptor wallets are both deterministic and operate in the hot/cold paradigm, which was first formalized by Das et al. (CCS 2019) for standard signature schemes. We introduce a new cryptographic primitive called *adaptor signatures with rerandomizable keys*, and use it to generically construct adaptor wallets. We further show how to instantiate adaptor signatures with rerandomizable keys from the ECDSA signature scheme and discuss that they can likely be built for Schnorr and Katz-Wang schemes as well. Finally, we discuss the limitations of the existing ECDSA- and Schnorr-based adaptor signatures w.r.t. deterministic wallets in the hot/cold setting and prove that it is impossible to overcome these drawbacks given the current state-of-the-art design of adaptor signatures.

1 Introduction

Blockchains have gained huge popularity in the past decade as they provide a decentralized infrastructure that allows not only to make simple payments but also to execute applications in a secure way. However, most Blockchains, including Bitcoin, only support the execution of simple applications while others, such as Monero or Zcash, are even more restrictive in their functionality and only support simple payments [20, 25]. Nevertheless, virtually all Blockchains rely on digital signatures in order to authenticate the origin of a transaction. While the functionality of Blockchains can be extended by appropriately adjusting the mining algorithms, this requires a hard fork of the Blockchain code which can take several years to complete in practice. In order to improve the restricted functionality of many Blockchains without having to change the Blockchain implementation and to allow for the execution of a larger class of applications, a new type of signature scheme called *adaptor signatures* was introduced by the cryptocurrency community [19] and first formally analyzed by Aumayr et al. [3]. At a high level, adaptor signatures allow two parties, say a *signer* and a *publisher* to trade a signature in exchange for a secret, i.e., if the publisher publishes a signature under the signer's secret key on the Blockchain, a secret value is leaked to the signer. More concretely, the publisher first generates an instance of a hard relation, i.e., a statement and witness pair and sends the statement to the signer. Using its secret key and the statement, the signer generates an incomplete signature called *pre-signature* which can be *adapted* by the publisher to a full valid signature using the witness. Once the adapted full signature is published, the signer can *extract* the witness given the pre- and full signature.

Adaptor signatures have proven to be extremely versatile for Blockchain applications. They allow for efficient constructions of two important categories of applications, namely payment channels (e.g., [3, 22]) and atomic swaps (e.g., [7, 24]), while requiring only a minimal functionality from the underlying Blockchain. Payment channels are a so-called off-chain solution, which allows two parties to issue many micropayments to each other without incurring fees for each transaction. Atomic swaps, on the other hand, allow two (or more) parties to atomically exchange tokens, i.e., either the exchange terminates and both parties obtain the other party's token or none does. Both of these applications rely on a technique that allows exchanging a secret value for a signature, which is exactly the functionality that adaptor signatures provide.

As the security of a user's funds in a Blockchain network depends solely on the secure storage of this user's signing secret key (and witnesses of adaptor signatures), it is of utmost importance how users store these secret values. Unfortunately, despite the increasing popularity of adaptor signatures, no prior work tried to address this issue. In other words we would like to answer the following question:

How can parties in practice employ adaptor signatures securely?

A concept known as *cryptographic wallets* has been introduced to use standard signature schemes securely in Blockchain networks. However, it has never been investigated if this concept can be extended to adaptor signatures.

Deterministic Wallets. One of the most promising proposals for cryptographic wallets are so-called deterministic wallets, which at a high level store a master signing key pair from which session key pairs are deterministically derived. Das et al. [6] gave the first formalization of such deterministic wallets in the hot/cold setting and later extended their model [5] to incorporate hierarchical wallets. In a bit more detail, a wallet scheme in the hot/cold setting consists of two separate devices, a hot and a cold wallet, that store the public and secret key respectively. The cold wallet is kept mostly offline and is only used to generate a new signature, whereas the hot wallet is constantly online to receive new transactions. This wallet structure ensures that it is inherently difficult for an attacker to steal the wallet's secret key, as it is stored in the offline cold wallet. Besides a standard unforgeability notion, wallet schemes should typically also satisfy an *unlinkability* property, which ensures that a third party cannot link two transactions issued to the same wallet. A naïve approach to achieve unlinkability is to let the wallet generate a fresh key pair for each transaction. This, however, requires the wallet to store all key pairs, which is not efficient, especially since cold wallets sometimes require special hardware (with limited storage) to securely store the secret keys. As such, deterministic wallets were introduced where the unlinkable keys are deterministically derived from a master key pair. This allows the wallet to derive new keys on the fly when they are needed instead of storing them indefinitely.

To date deterministic wallets have only been analyzed for digital signature schemes (e.g., [6]). Considering that the security of adaptor signatures does not only depend on the secure storage of the secret key but also on the secure handling of witnesses, designing a secure wallet scheme for adaptor signatures becomes even more pressing.

1.1 Our Contribution

In this work, we initiate the study of deterministic wallets in the hot/cold setting for adaptor signatures following the approach of Das et al. [6]. To this end, we first introduce a new notion of adaptor signatures, which we call *adaptor signature with rerandomizable keys*. This primitive extends regular adaptor signatures by key rerandomization algorithms. That is, given an adaptor signature key pair (sk, pk) and some randomness ρ, an adaptor signature with rerandomizable keys allows to deterministically and independently rerandomize sk and pk using ρ to obtain a new key pair (sk', pk') such that (1) (sk', pk') constitutes a valid signing key pair, and (2) (sk', pk') is indistinguishable from a freshly generated key pair. We formally define this primitive and show how to instantiate it by transforming the existing ECDSA-adaptor signature scheme [3, 18] into an adaptor signature with rerandomizable keys.

We provide a formal model for adaptor wallets (in the full version of this paper[1]). Our adaptor wallets are the first cryptographic wallets that are deterministic, in the hot/cold setting and support the use of adaptor signatures. While the hot/cold wallet setting allows to provide strong security guarantees, it is not suitable for all applications in practice. Payment channels, for instance, have a short life span but require a frequent exchange of signatures. As such, storing the secret key in an offline cold wallet seems counterintuitive. Instead, for such applications our model allows to store secret values on one online device while guaranteeing that

[1] The full version will be published on the IACR Cryptology ePrint Archive.

even if this device gets corrupted, the master key pair and other keys derived from the master key pair remain secure. To achieve this feature, we use the idea of hardened/non-hardened wallets as defined in [5] and adjust it for adaptor wallets (see Section 4.1 for more details).

We then show how to generically construct adaptor wallets from *any* adaptor signature scheme with rerandomizable keys where the hard relation is *witness rerandomizable* and further show how to initiate such a relation for ECDSA-adaptor signatures. Witness rerandomizability of a hard relation R essentially means that for any statement/witness pair $(Y, y) \in R$ the witness y can be rerandomized deterministically using some randomness ρ to a witness y' with corresponding statement Y' such that $(Y', y') \in R$. We require this property to alleviate the storage constraints on the cold wallet, i.e., as explained above, the cold wallet is often a storage restricted device and hence deterministic rerandomization can be useful to generate required values on the fly instead of storing them long-term. Although we do not formally show how adaptor wallets can be instantiated from Schnorr and Katz-Wang signature schemes [21, 13], it seems that our approach can be used in order to transform these schemes to adaptor signatures with rerandomizable keys and use them to instantiate adaptor wallets.

Our final contribution is closely related to witness rerandomizable hard relations. Surprisingly, we show that it is *impossible* to construct an adaptor wallet from fully rerandomizable hard relations, i.e., hard relations where the statement and witness can be rerandomized independently using the same randomness. This is in stark contrast to the secret and public keys which can be rerandomized independently. We believe that our work paves the way for mainstreaming the usage of adaptor signatures by providing a secure and efficient deterministic wallet framework in the hot/cold setting.

1.2 Related Work

We divide the related work into adaptor signatures and deterministic wallets.

Adaptor Signatures. After being first introduced by Poelstra [19], adaptor signatures have been used in many Blockchain related applications, such as atomic swaps [7], payment channel networks [17] and payment channel hubs [22]. Aumayr et al. [3] later provided a standalone formalization of this primitive. Shortly after, Esgin et al. and Tairi et al. [9, 23] provided instantiations of adaptor signatures in the post-quantum setting where the adversary has access to a quantum computer while the end users do not. Finally, Erwig et al. [8] showed how to generically transform signature schemes built from identification schemes which satisfy certain properties, into single party and two party adaptor signatures. There have been several other recent works on adaptor signatures (e.g., [16, 24]) which used or extended this primitive to build more complex applications.

Deterministic Wallets. There have been many recent works formalizing and analyzing cryptographic wallets, such as [12, 15, 2, 14]. The concept of deterministic wallets in the hot/cold setting was first formalized and instantiated by Das et al. [6]. Alkadri et al. [1] later showed how to realize such wallets with security in the post-quantum setting. In a follow-up work, Das et al. [5] extended the original model by allowing hierarchical derivation of new wallets. In order to guarantee security even in case one of such wallets is corrupted, e.g., when a wallet is not implemented in the hot/cold setting, the authors introduced two different key derivation mechanisms, namely hardened key derivation for keys that might be leaked to the adversary and non-hardened key derivation for keys that are stored securely via the hot/cold wallet paradigm. Later, Yin et al. [26] introduced hierarchical deterministic wallets that support stealth addresses. However, none of these works have considered adaptor signature support for deterministic wallets.

2 Preliminaries

Notation. We denote by $s \leftarrow_\$ H$ the uniform random sampling of a value s from the set H. For an integer l, the notation $[l]$ denotes the set of integers $\{1, \cdots, l\}$ and for a randomized algorithm A, we denote by $y \leftarrow_\$ A(x)$ the execution of A on input x that outputs y. For a deterministic algorithm B, we write $y \leftarrow B(x, \rho)$ to denote the execution of B on input x and ρ that outputs y. By $y \in A(x)$ we denote that y is an element in the set of possible outputs of an execution of A on input x. Throughout our paper, we assume that public parameters par can be used as input to all algorithms. For two strings a and b, we write $a = (b, \cdot)$ if b is a prefix of a. We abbreviate the expressions *deterministic polynomial time* and *probabilistic polynomial time* by DPT and PPT respectively.

2.1 Non-interactive zero-knowledge proofs

A non-interactive zero knowledge proof (NIZK) [4] with respect to a polynomial-time recognizable binary relation R is given by the following tuple of algorithms $\mathsf{NIZK} := (\mathsf{Setup}_R, \mathsf{P}, \mathsf{V})$, where (i) $\mathsf{Setup}_R(1^n)$ outputs a common reference string crs; (ii) $\mathsf{P}(\mathsf{crs}, (Y, y))$ outputs a proof π for $(Y, y) \in R$; (iii) $\mathsf{V}(\mathsf{crs}, Y, \pi)$ outputs a bit $b \in \{0, 1\}$. Further, the NIZK proof of knowledge w.r.t. R should satisfy the properties *completeness*, *soundness*, and *zero knowledge*. We do not go into the details of these properties here.

2.2 (Witness rerandomizable) Hard relation.

Definition 1 (Hard Relation). *Let $R \subseteq \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ be a relation with statement/witness pairs $(Y, y) \in \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ and let the language $L_R \subseteq \mathcal{D}_{\mathsf{Y}}$ associated to R be defined as $L_R := \{Y \in \mathcal{D}_{\mathsf{Y}} \mid \exists y \in \mathcal{D}_{\mathsf{w}} \text{ s.t. } (Y, y) \in R\}$. We say that R is a* hard relation *if: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^n)$ that on input the security parameter outputs a pair $(Y, y) \in R$; (ii) There exists a PPT algorithm $\mathsf{WitToSt}(y)$ that on input a witness y outputs a statement Y, s.t. $(Y, y) \in R$; (iii) The relation R is poly-time decidable; (iv) For all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a valid witness $y \in \mathcal{D}_{\mathsf{w}}$ for $Y \in L_R$ is negligible.*

In this work we require a stronger notion of hard relation namely hard relations that are witness rerandomizable.

Definition 2 (Witness Rerandomizable Hard Relation). *Let $R \subseteq \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ be a hard relation with statement/witness pairs $(Y, y) \in \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ and let the public parameters* par *define a randomness space $X := X(\mathsf{par})$. Further, let* $\mathsf{RandWit}$ *be a DPT algorithm which is defined as follows:*
$\mathsf{RandWit}(y, \rho)$*: The deterministic witness randomization algorithm takes as input a witness $y \in \mathcal{D}_{\mathsf{w}}$, a randomness $\rho \in X$ and outputs a rerandomized witness y'.*

We say that R is perfectly witness rerandomizable *if for all $(\cdot, y) \in \mathsf{GenR}(1^n)$ and all $\rho \leftarrow_{\$} X$ the distributions of (Y', y') and (Y'', y'') are identical, where:*

$$\begin{aligned}(Y', y') &\leftarrow (\mathsf{WitToSt}(\mathsf{RandWit}(y, \rho)), \mathsf{RandWit}(y, \rho))\\ (Y'', y'') &\leftarrow \mathsf{GenR}(1^n)\end{aligned}$$

2.3 Adaptor Signatures

We recall the definition of an adaptor signature scheme by Aumayr et al. [3].

Definition 3 (Adaptor signature scheme). *An adaptor signature scheme w.r.t. a hard relation R and a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ consists of four algorithms $\mathsf{ASig}_{R,\Sigma} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ with the following syntax:* $\mathsf{pSign}(sk, m, Y)$ *is a PPT algorithm that on input a secret key sk, message $m \in \{0, 1\}^*$ and statement $Y \in L_R$, outputs a pre-signature $\tilde{\sigma}$;* $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma})$ *is a DPT algorithm that on input a public key pk, message $m \in \{0, 1\}^*$, statement $Y \in L_R$ and pre-signature $\tilde{\sigma}$, outputs a bit b;* $\mathsf{Adapt}(\tilde{\sigma}, y)$ *is a DPT algorithm that on input a pre-signature $\tilde{\sigma}$ and witness y, outputs a signature σ; and* $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$ *is a DPT algorithm that on input a signature σ, pre-signature $\tilde{\sigma}$ and statement $Y \in L_R$, outputs a witness y such that $(Y, y) \in R$, or $\perp$.*

An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ must satisfy *pre-signature correctness* stating that for every $m \in \{0, 1\}^*$ and every $(Y, y) \in R$, the following holds:

$$\Pr\left[\begin{array}{l|l}\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma}) = 1, & (sk, pk) \leftarrow \mathsf{Gen}(1^n),\ \tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)\\ \mathsf{Verify}(pk, m, \sigma) = 1, (Y, y') \in R & \sigma := \mathsf{Adapt}_{pk}(\tilde{\sigma}, y),\ y' := \mathsf{Ext}(pk, \sigma, \tilde{\sigma}, Y)\end{array}\right] = 1.$$

An adaptor signature scheme has to satisfy the following properties.

```
$\mathsf{aWitExt}(n)$
$\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(m^*, Y^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathtt{SignO}(\cdot), \mathtt{PreSignO}(\cdot,\cdot)}(pk)$
$\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk, m^*, Y^*)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathtt{SignO}, \mathtt{PreSignO}}(\tilde{\sigma}^*, \mathsf{st})$
$b_1 \leftarrow (Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, Y^*)) \notin R$
$b_2 \leftarrow m^* \notin \mathcal{Q}$
$b_3 \leftarrow \mathsf{Verify}(pk, m^*, \sigma^*)$
$b_4 \leftarrow Y^* \in L_R$
Return $(b_1 \wedge b_2 \wedge b_3 \wedge b_4)$

Oracle $\mathtt{PreSignO}(m, Y)$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
Return $\tilde{\sigma}$

$\mathsf{aSigForge}(n)$
$\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(Y, y) \leftarrow \mathsf{GenR}(1^n)$
$(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathtt{SignO}, \mathtt{PreSignO}}(pk, Y)$
$\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk, m^*, Y)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathtt{SignO}, \mathtt{PreSignO}}(\tilde{\sigma}^*, \mathsf{st})$
Return
$(m^* \notin \mathcal{Q} \wedge \mathsf{Verify}(pk, m^*, \sigma^*))$

Oracle $\mathtt{SignO}(m)$
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
Return $\sigma$
```

Fig. 1. $\mathsf{aSigForge}$ and $\mathsf{aWitExt}$ games for an adaptor signature scheme ASig.

Definition 4 (Existential unforgeability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is unforgeable if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function ν such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$, *where the experiment $\mathsf{aSigForge}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}$ is defined as in Fig. 1.*

Definition 5 (Pre-signature adaptability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ satisfies pre-signature adaptability if for any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any public key pk and any pre-signature $\tilde{\sigma} \in \{0,1\}^*$ with $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma}) = 1$, we have $\mathsf{Verify}(pk, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1$.*

Definition 6 (Witness extractability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is* witness extractable *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$, *where the experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}$ is defined as in Fig. 1.*

Definition 7. *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is secure, if it is unforgeable, pre-signature adaptable and witness extractable.*

2.4 ECDSA-based Adaptor Signature

We briefly recall the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}] = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ as presented by Aumayr et al. [3], which is defined w.r.t. the positive ECDSA signature scheme $\mathsf{PEC} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a hard relation R_g. Recall that the positive ECDSA scheme operates over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p and that the key generation outputs a key pair (sk, pk) with $sk \leftarrow_{\$} \mathbb{Z}_p$ and $pk \leftarrow g^{sk}$. A message $m \in \{0,1\}^*$ is then signed by first sampling $k \leftarrow_{\$} \mathbb{Z}_p$, setting $r \leftarrow f(g^k)$ and computing $s := k^{-1}(\mathsf{H}(m) + r \cdot sk)$, where $\mathsf{H} : \{0,1\} \rightarrow \mathbb{Z}_p$ is a hash function and $f : \mathbb{G} \rightarrow \mathbb{Z}_p$. The signature is then $\sigma := (r, s)$, which can be verified by checking if $f(g^{s^{-1}\mathsf{H}(m)} pk^{s^{-1}r}) = r$. The hard relation R_g is defined as $R_g := \{((Y, \pi), y) | Y = g^y \wedge \mathsf{V}(Y, \pi) = 1\}$, i.e., it is the standard dlog relation with an additional non-interactive zero knowledge (NIZK) proof, which proves knowledge of the witness. The additional NIZK proof is required for technical reasons which we do not discuss here. Apart from the NIZK proof for relation R_g, the $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ construction also includes a NIZK proof for another relation $R_Y := \{((\tilde{K}, K), k) | \tilde{K} = g^k \wedge K = Y^k\}$. For further details we refer to [3]. The construction of $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ is depicted in Fig. 2.

3 Adaptor Signatures with Rerandomizable Keys

In this section we define the notion of adaptor signatures with rerandomizable keys and show how to instantiate it. Later in Sec. 4.1 we will use this primitive to generically construct adaptor wallets.

```
$\mathsf{pSign}(sk, m, I_Y)$
    $x := sk, (Y, \pi_Y) := I_Y$
    $k \leftarrow_\$ \mathbb{Z}_q, \tilde{K} := g^k$
    $K := Y^k, r := f(K)$
    $\tilde{s} := k^{-1}(\mathsf{H}(m) + rx)$
    $\pi \leftarrow \mathsf{P}_Y((\tilde{K}, K), k)$
    **return** $(r, \tilde{s}, K, \pi)$
```

```
$\mathsf{pVrfy}(pk, m, I_Y, \tilde{\sigma})$
    $X := pk, (Y, \pi_Y) := I_Y$
    $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $u := \mathsf{H}(m) \cdot \tilde{s}^{-1}$
    $v := r \cdot \tilde{s}^{-1}, K' := g^u X^v$
    **return** $(I_Y \in L_R$
        $\wedge (r = f(K)) \wedge \mathsf{V}_Y((K', K), \pi))$
```

```
$\mathsf{Adapt}(\tilde{\sigma}, y)$
    $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $s := \tilde{s} \cdot y^{-1}$
    **return** $(r, s)$
```

```
$\mathsf{Ext}(\sigma, \tilde{\sigma}, I_Y)$
    $(r, s) := \sigma$
    $(\tilde{r}, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $y' := s^{-1} \cdot \tilde{s}$
    **if** $(I_Y, y') \in R_g$
        **then return** $y'$
    **else return** $\perp$
```

Fig. 2. ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ instantiated with a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$.

3.1 Definition

The notion of signature schemes with rerandomizable keys has first been introduced by Fleischhacker et al. [11] and has since been proven to be useful for the construction of deterministic wallet schemes (e.g., [6, 5]). Essentially, a signature scheme with rerandomizable keys extends regular signature schemes by two deterministic algorithms, a public key and a secret key rerandomization algorithm, which on input a public key or a secret key respectively and a randomness, output rerandomized keys. Such keys, if rerandomized with the same randomness, constitute a new signing key pair, which is distributed identically to a freshly and independently generated signing key pair. These properties and the deterministic nature of the rerandomization make such signature schemes good candidates for the construction of deterministic wallets. In our work, we are concerned with adaptor signatures. Therefore, we define in the following the notion of adaptor signatures with rerandomizable keys.

Definition 8 (Adaptor signature scheme with rerandomizable keys). *An adaptor signature scheme with rerandomizable keys w.r.t. a hard relation R and a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ consists of six algorithms $\mathsf{RASig}_{R,\Sigma} = (\mathsf{RandSK}, \mathsf{RandPK}, \mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ where $(\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ are the same algorithms as defined for adaptor signatures (cf. Def. 3). Assuming that the public parameters* par *define a randomness space $X := X(\mathsf{par})$, the remaining algorithms are defined as follows:*

$\mathsf{RandSK}(sk, \rho)$*: The deterministic secret key rerandomization algorithm takes as input a secret key sk and a randomness $\rho \in X$ and outputs a rerandomized secret key sk'.*

$\mathsf{RandPK}(pk, \rho)$*: The deterministic public key rerandomization algorithm takes as input a public key pk and a randomness $\rho \in X$ and outputs a rerandomized public key pk'.*

An adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$ must satisfy the following two correctness properties:

1. Pre-signature correctness *stating that for all $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$, all $m \in \{0,1\}^*$, all $\rho \in X$ and all $(Y, y) \in R$, the rerandomized keys $sk' \leftarrow \mathsf{RandSK}(sk, \rho)$ and $pk' \leftarrow \mathsf{RandPK}(pk, \rho)$ satisfy:*

$$\Pr\left[\begin{matrix}\mathsf{pVrfy}(pk', m, Y, \tilde{\sigma}) = 1, \\ \mathsf{Verify}(pk', m, \sigma) = 1, (Y, y') \in R\end{matrix} \middle| \begin{matrix}\tilde{\sigma} \leftarrow \mathsf{pSign}(sk', m, Y), \\ \sigma := \mathsf{Adapt}(\tilde{\sigma}, y),\end{matrix}\ y' := \mathsf{Ext}(\sigma, \tilde{\sigma}, Y)\right] = 1.$$

2. (Perfect) rerandomizability of keys: *For all $(sk, pk) \in \mathsf{Gen}(1^n)$ and $\rho \leftarrow_\$ X$, the distributions of (sk', pk') and (sk'', pk'') are identical, where:*

$$(sk', pk') \leftarrow (\mathsf{RandSK}(sk, \rho), \mathsf{RandPK}(pk, \rho)),$$
$$(sk'', pk'') \leftarrow_\$ \mathsf{Gen}(1^n).$$

Like adaptor signatures, an $\mathsf{RASig}_{R,\Sigma}$ scheme must satisfy pre-signature adaptability.

Definition 9 (Pre-signature adaptability). *An adaptor signature scheme with perfectly rerandomizable keys* $\mathsf{RASig}_{R,\Sigma}$ *satisfies pre-signature adaptability if for any message* $m \in \{0,1\}^*$*, any statement/witness pair* $(Y, y) \in R$*, any public key* pk *and any pre-signature* $\tilde{\sigma} \in \{0,1\}^*$ *with* $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma}) = 1$*, we have* $\mathsf{Verify}(pk, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1$.

For adaptor signatures with rerandomizable keys, we introduce the notions of *existential unforgeability under honestly rerandomizable keys* and *witness extractability under honestly rerandomizable keys*. These notions extend the respective security notions of adaptor signatures by allowing the adversary to not only obtain (pre-)signatures under sk but also under secret keys that constitute honest rerandomizations of sk. An honest rerandomization is one where the randomness has been chosen uniformly at random from the randomness space X. Further, in our security notions the adversary can win the game by providing a forgery either under sk or under any honestly rerandomized key. We formally describe these security notions in Fig. 3.

Definition 10 (Existential unforgeability under honestly rerandomizable keys). *An adaptor signature scheme with rerandomizable keys* $\mathsf{RASig}_{R,\Sigma}$ *is unforgeable if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}$ *is defined as in Fig. 3.*

Definition 11 (Witness extractability under honestly rerandomizable keys). *An adaptor signature scheme with rerandomizable keys* $\mathsf{RASig}_{R,\Sigma}$ *is witness extractable if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* ν *such that the following holds:* $\Pr[\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}$ *is defined as in Fig. 3.*

```
$\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n)$
$\mathcal{Q} := \emptyset, \mathcal{R} := \emptyset$
$(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(Y, y) \leftarrow \mathsf{GenR}(1^n)$
$(m^*, \rho^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk, Y)$
$sk^* \leftarrow \mathsf{RandSK}(sk, \rho^*)$
$pk^* \leftarrow \mathsf{RandPK}(pk, \rho^*)$
$\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk^*, m^*, Y)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\tilde{\sigma}^*, \mathsf{st})$
$b_1 \leftarrow m^* \notin \mathcal{Q}$
$b_2 \leftarrow \mathsf{Verify}(pk^*, m^*, \sigma)$
$b_3 \leftarrow \rho^* \in \mathcal{R}$
Return $(b_1 \wedge b_2 \wedge b_3)$
```

```
$\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n)$
$\mathcal{Q} := \emptyset, \mathcal{R} := \emptyset$
$(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(m^*, \rho^*, Y^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk)$
$sk^* \leftarrow \mathsf{RandSK}(sk, \rho^*)$
$pk^* \leftarrow \mathsf{RandPK}(pk, \rho^*)$
$\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk^*, m^*, Y^*)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\tilde{\sigma}^*, \mathsf{st})$
$b_1 \leftarrow (Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, Y^*)) \notin R$
$b_2 \leftarrow m^* \notin \mathcal{Q}$
$b_3 \leftarrow \mathsf{Verify}(pk^*, m^*, \sigma^*)$
$b_4 \leftarrow \rho^* \in \mathcal{R}$
$b_5 \leftarrow Y^* \in L_R$
Return $(b_1 \wedge b_2 \wedge b_3 \wedge b_4 \wedge b_5)$
```

```
Oracle RSignO$(m, \rho)$
If $\rho \notin \mathcal{R}$ : return 0
$sk' \leftarrow \mathsf{RandSK}(sk, \rho)$
$\sigma \leftarrow \mathsf{Sign}(sk', m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
Return $\sigma$

Oracle PreSignO$(m, Y, \rho)$
If $\rho \notin \mathcal{R}$ : return 0
$sk' \leftarrow \mathsf{RandSK}(sk, \rho)$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk', m, Y)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
Return $\tilde{\sigma}$

Oracle RandO
$\rho \leftarrow_{\$} X$
$\mathcal{R} := \mathcal{R} \cup \{\rho\}$
Return $\rho$
```

Fig. 3. $\mathsf{aSigForge{-}hrk}$ and $\mathsf{aWitExt{-}hrk}$ games for an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$. In the above games we have $\mathcal{O} := \{\mathtt{RSignO}, \mathtt{PreSignO}, \mathtt{RandO}\}$.

3.2 Construction

In Fig. 4, we present an adaptor signature with rerandomizable keys $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}]$ from the ECDSA-based adaptor signature $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ from Fig. 2. Similar to the rerandomizable ECDSA construction of Das et al. [6], we use public key-prefixed messages in our construction which is required to ensure security (see the proof sketch of Thm. 1) and we use a hash function $\mathsf{H} \colon \{0,1\}^* \to \mathbb{Z}_p$.

To prove the security of our construction, we follow the approach of Das et al. [6], who presented a security proof of the plain ECDSA signature scheme with rerandomizable keys via a reduction to the (non-rerandomizable) ECDSA signature scheme. The main ingredient in their security proof is a related key attack

```
Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{pSign}\,(sk, m, Y)$
00 $\mathsf{pm} \leftarrow (pk, m)$
01 $\tilde{\sigma} \leftarrow \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{pSign}\,(sk, \mathsf{pm}, Y)$
02 Return $\tilde{\sigma}$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{pVrfy}\,(pk, m, Y, \tilde{\sigma})$
03 $\mathsf{pm} \leftarrow (pk, m)$
04 Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{pVrfy}\,(pk, \mathsf{pm}, Y, \tilde{\sigma})$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Adapt}\,(\tilde{\sigma}, y)$
05 Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Adapt}\,(\tilde{\sigma}, y)$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{RandSK}\,(sk, \rho)$
06 $sk' \leftarrow sk \cdot \rho \bmod p$
07 Return $sk'$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Sign}\,(sk, m)$
08 $\mathsf{pm} \leftarrow (pk, m)$
09 $\sigma \leftarrow \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Sign}\,(sk, \mathsf{pm})$
10 Return $\sigma$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Verify}\,(pk, \sigma, m)$
11 $\mathsf{pm} \leftarrow (pk, m)$
12 Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Verify}\,(pk, \sigma', \mathsf{pm})$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Ext}\,(\sigma, \tilde{\sigma}, Y)$
13 Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Ext}\,(\sigma, \tilde{\sigma}, Y)$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{RandPK}\,(pk, \rho)$
14 $pk' \leftarrow pk^{\rho}$
15 Return $pk'$
```

Fig. 4. Construction of a key-prefixed ECDSA-based adaptor signature scheme with perfectly rerandomizable keys $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}]$ from the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ as described in Fig. 2. Both schemes are instantiated with a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$.

which allows to transform a signature on message m_1 under public key pk_1 to a signature for message m_0 under a related public key pk_0. We recall their transformation in the following (and formally in Lemma 1 and Fig. 5).

Let $\mathsf{PEC}[\mathsf{H}_0]$ and $\mathsf{PEC}[\mathsf{H}_1]$ denote two (positive) ECDSA signature schemes instantiated with hash functions H_0 and H_1 respectively. Then the authors show that if $pk_1 = (pk_0)^{\rho}$ where $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$ and given a valid signature σ_1 (i.e., $\mathsf{PEC}[\mathsf{H}_1].\mathsf{Verify}(pk_1, m_1, \sigma_1) = 1$), the algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$ returns a valid signature σ_0 under pk_0 and m_0, i.e., $\mathsf{PEC}[\mathsf{H}_0].\mathsf{Verify}(pk_0, m_0, \sigma_0) = 1$. For this transformation, Das et al. state and prove the following lemma.

Lemma 1. *Consider the algorithm* $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ *in Figure 5. Suppose that:*

– $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$, $pk_0, pk_1 \in \mathbb{G}$ *s.t.* $pk_0 = g^{x_0}$ *and* $pk_1 = pk_0^{\rho}$,
– $\mathsf{PEC}[\mathsf{H}_1].\mathsf{Verify}(pk_1, m_1, \sigma_1) = 1$, $\sigma_0 \leftarrow \mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$.

Then $\mathsf{PEC}[\mathsf{H}_0].\mathsf{Verify}(pk_0, m_0, \sigma_0) = 1$.

```
$\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$
00 $z_0 \leftarrow \mathsf{H}_0(m_0)$
01 $z_1 \leftarrow \mathsf{H}_1(m_1)$
02 If $\big(\mathsf{PEC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Verify}(pk_1, \sigma_1, m_1) = 0\big)$
   $\vee \Big(\rho \neq \frac{z_1}{z_0} \vee pk_1 \neq pk_0^{\rho}\Big)$ :
03     Return $\perp$
04 $(r, s_1) \leftarrow \sigma_1$
05 $s_0 \leftarrow \frac{s_1}{\rho} \bmod p$
06 $\sigma_0 \leftarrow (r, s_0)$
07 Return $\sigma_0$

$\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$
00 $z_0 \leftarrow \mathsf{H}_0(m_0)$
01 $z_1 \leftarrow \mathsf{H}_1(m_1)$
02 If $\big(\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1)\vee$
   $\Big(\rho \neq \frac{z_1}{z_0} \vee pk_1 \neq pk_0^{\rho} \vee I_Y \notin L_R\Big)$ :
03     Return $\perp$
04 $(r, \tilde{s}_1, K, \pi) \leftarrow \tilde{\sigma}_1$
05 $\tilde{s}_0 \leftarrow \frac{\tilde{s}_1}{\rho} \bmod p$
06 $\tilde{\sigma}_0 \leftarrow (r, \tilde{s}_0, K, \pi)$
07 Return $\tilde{\sigma}_0$
```

Fig. 5. Figure shows the $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ and $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ algorithms for hash functions $\mathsf{H}_0, \mathsf{H}_1\colon \{0,1\}^* \to \mathbb{Z}_p$.

We show that a similar transformation can be applied to the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ to transform pre-signatures. Since pre-signatures in this scheme include a zero-knowledge proof, it is not immediately clear that such a transformation goes through. We next give the lemma for the pre-signature transformation as well as the proof for the lemma.

Lemma 2. *Let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *and* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *denote two ECDSA-based adaptor signature schemes according to Fig. 2 instantiated with hash functions* H_0 *and* H_1*. Consider the algorithm* $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ *in Figure 5. Suppose that:*

- $I_Y \in L_{R_Y}$, $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$,
- $pk_0, pk_1 \in \mathbb{G}$ *s.t.* $pk_0 = g^{x_0}$ *and* $pk_1 = pk_0^{\rho}$,
- $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1) = 1$,
- $\tilde{\sigma}_0 \leftarrow \mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$.

Then $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, I_Y, \tilde{\sigma}_0) = 1$.

We would like to point out that Lemma 2 requires that after the transformation, the new pre-signature $\tilde{\sigma}_0$ is indeed valid with respect to the same statement I_Y. In other words, given the witness y, both $\tilde{\sigma}_0$ and $\tilde{\sigma}_1$ can be adapted into full signatures under pk_0 and pk_1 respectively.

Proof. The proof of this lemma is similar to the proof of Lemma 1 from [6]. To prove the lemma, we have to show that given a statement $I_Y := (Y, \pi_Y) \in L_R$, a public key $pk_1 = pk_0^{\rho}$ where $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)}$ and a pre-signature $\tilde{\sigma}_1$ such that $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1) = 1$, $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ outputs a pre-signature $\tilde{\sigma}_0$ such that $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, I_Y, \tilde{\sigma}_0) = 1$. Recall that for the pre-signature $\tilde{\sigma}_1 := (r, \tilde{s}_1, K, \pi)$ it holds that $\tilde{s}_1 = k^{-1}(\mathsf{H}_1(m) + r \cdot sk_1)$, $r := f(K)$, $K := Y^k$ and π is a valid proof that $(\tilde{K}, K)$ is a valid statement in R_Y. Then $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, Y, \tilde{\sigma}_0)$ computes the following:

$$\begin{aligned} K' =& g^u \cdot pk_0^v = g^{(\mathsf{H}_0(m_0) \cdot \tilde{s}_0^{-1})} \cdot pk_0^{r \cdot \tilde{s}_0^{-1}} = g^{\tilde{s}_0^{-1} \cdot (\mathsf{H}_0(m_0) + x_0 \cdot r)} \\ =& g^{\frac{\rho}{\tilde{s}_1} \cdot (\mathsf{H}_1(m_1) \cdot \rho^{-1} + x_1 \cdot \rho^{-1} \cdot r)} = g^{\frac{\rho}{k^{-1}(\mathsf{H}_1(m_1) + x_1 \cdot r)} \cdot (\mathsf{H}_1(m_1) + x_1 \cdot r) \cdot \rho^{-1}} = g^{\frac{\rho \cdot \rho^{-1}}{k^{-1}}} = g^k \end{aligned}$$

Therefore, the zero-knowledge proof π is valid w.r.t. the statement (K', K) where $K' = g^k$ and $K = Y^k$. We can conclude that the pre-signature $\tilde{\sigma}_0 \leftarrow \mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$ with $\tilde{\sigma}_0 := (r, \frac{\tilde{s}_1}{\rho}, K, \pi)$ constitutes a valid pre-signature w.r.t. public key pk_0, message m_0 and statement I_Y.

Theorem 1. *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions modeled as random oracle and let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *be the secure ECDSA-based adaptor signature as per Fig. 2. Then the construction* $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *as described in Fig. 4 is existentially unforgeable under honestly rerandomizable keys as per Def. 10.*

Proof (Sketch). The proof of this theorem is similar to the proof of the multiplicatively rerandomizable ECDSA signature scheme as provided by Das et al. [6]. In their proof, the authors show unforgeability of an ECDSA scheme with rerandomizable keys by exhibiting a reduction to the unforgeability of the regular ECDSA signature scheme. The proof of Das et al. relies crucially on the related key attack as depicted by the algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ in Fig. 5, which allows to transform a signature under a public key pk to a valid signature under a related public key $pk' \leftarrow pk^{\rho'}$, if ρ' has a certain structure. In more details, Das et al. instantiate the ECDSA scheme with a hash function H_0 and the ECDSA scheme with rerandomizable keys with a hash function H_1 (both hash functions are modeled as random oracles). They then program the random oracle H_1 in such a way that on input $m' = (pk', m)$, where pk' is a public key rerandomized with randomness ρ' (i.e., $pk' = pk^{\rho'}$), it holds $\mathsf{H}_1(m') = \mathsf{H}_0(m) \cdot \rho'$. This allows the reduction to transform signatures for rerandomized public keys to signatures for the original public key and vice versa using algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$.

In our proof, we can show unforgeability of the $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ scheme via a reduction to the unforgeability of the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. The main difference in our proof as compared to the proof of Das et al. arises from the fact that we need to apply the related key attack on pre-signatures as well. This transformation requires us to use the algorithm $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ as described in Fig. 5. To apply this transformation, we program the random oracle H_1 in exactly the same way as is done in the proof of Das et al. and hence, the programming of H_1 is consistent for signatures and pre-signatures.

Theorem 2. *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions modeled as random oracle and let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *be the secure ECDSA-based adaptor signature as per Fig. 2. Then the construction* $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *as described in Fig. 4 is witness extractable under honestly rerandomizable keys as per Def. 11.*

Proof (Sketch). The proof of this theorem is similar to the proof of Thm. 1. Here we must provide a reduction to the witness extractability property $\mathsf{aWitExt}$ of the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. However, here we have to show that a valid forgery in game $\mathsf{aWitExt{-}hrk}$ for scheme $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ can be transformed into a valid forgery in game $\mathsf{aWitExt}$ for scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. Recall that for a valid forgery σ^* in game $\mathsf{aWitExt{-}hrk}$ and given the corresponding pre-signature $\tilde{\sigma}^*$, it must hold that $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$. Therefore, we must show that applying the transformations $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ and $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ from Fig. 5 on σ^* and $\tilde{\sigma}^*$ respectively preserves the above condition w.r.t. scheme $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. We show this via the following claim, for which we assume that $m_0, m_1 \in \{0,1\}^*$ are two messages, $\rho^* = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$ and $pk^* = pk_{\mathsf{aWitExt}}^{\rho^*}$, where pk_{aWitExt} is the public key in game $\mathsf{aWitExt}$.

Claim 1 *If it holds that* $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$ *then we have* $(I_Y^*, \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)) \notin R_g$*, where*

$$\sigma' \leftarrow \mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma^*, \rho^*, pk_{\mathsf{aWitExt}}, pk^*)$$

$$\tilde{\sigma}' \leftarrow \mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \tilde{\sigma}^*, \rho^*, pk_{\mathsf{aWitExt}}, pk^*, I_Y^*).$$

Let $\sigma^* = (r, s)$ and $\tilde{\sigma}^* = (r, \tilde{s}, K, \pi)$, then we have: $\sigma' := (r, \frac{s}{\rho^*}), \tilde{\sigma}' := (r, \frac{\tilde{s}}{\rho^*}, K, \pi)$. Therefore, we can conclude that:

$$\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*) = s^{-1}\tilde{s} = \left(\frac{s}{\rho^*}\right)^{-1} \frac{\tilde{s}}{\rho^*} = \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)$$

Hence, we can conclude that if $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$ then $(I_Y^*, \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)) \notin R_g$. And thus, a forgery in game $\mathsf{aWitExt{-}hrk}$ can be transformed into a valid forgery in game $\mathsf{aWitExt}$.

We note that pre-signature adaptability (cf. Def. 9) of $\mathsf{REC}_{R_g,\mathsf{PEC}}$ follows immediately from the pre-signature adaptability property of $\mathsf{EC}_{R_g,\mathsf{PEC}}$.

3.3 Discussion

Note that our ECDSA-based instantiation of an adaptor signature with rerandomizable keys is compatible with a plethora of cryptocurrencies, since many cryptocurrency networks, including Bitcoin and Ethereum, rely on the ECDSA signature scheme. In our instantiation, we use a multiplicative key rerandomization instead of an additive one. This seemingly insignificant difference has a crucial impact on the security of the resulting scheme as shown by Das et al. [5]. More concretely, Das et al. presented an ECDSA scheme with additive key rerandomization, which incurred a security loss in the number of rerandomized keys, whereas the ECDSA scheme with multiplicative rerandomization from [6] does not incur such a loss.[2]. In a nutshell, this security loss stems from the related key attack that is required to prove security of the additively rerandomizable scheme. Since the security proof for ECDSA-based adaptor signatures with rerandomizable keys would rely on the same related key attack, a similar security loss can be expected for the additively rerandomizable ECDSA-based adaptor signature. Worse yet, the related key attack for additively rerandomizable ECDSA allows to prove only *one-per-message unforgeability* [10], which is a weaker security notion than standard unforgeability. Therefore, we used multiplicative rerandomization in our instantiation.

While we did not work out the details, it is likely that adaptor signatures with rerandomizable keys can be constructed from Schnorr and Katz-Wang-based adaptor signatures [8] (due to the existing related key attack for Schnorr signatures as presented in [11]). Finally, we believe that it would be an interesting future work to extend the notion of two-party adaptor signatures as presented in [8] to two-party adaptor signatures with rerandomizable keys.

[2] Das et al. show that this loss results in 20 bits less security for certain parameters. We refer the reader to [5] for details.

4 Adaptor Wallets

In this section, we introduce the idea of adaptor wallets, which securely maintain and operate adaptor signature schemes in a cryptocurrency network. We first provide a high level overview of our model and then provide a generic wallet construction from any adaptor signature scheme with rerandomizable keys and witness rerandomizable hard relation. Finally, we show that it is impossible to achieve deterministic and independent statement/witness rerandomization in our model. Due to space limitations, we defer the full formal model and the security arguments for our generic construction to the full version of this paper.

4.1 Model and Construction

We now describe a model for adaptor wallets and we discuss how adaptor signature schemes with rerandomizable keys can be used to instantiate such a wallet. Our notion of adaptor wallets resembles the notion of hierarchical deterministic wallets by Das et al. [5], however, extending their notion to support adaptor signature operations such as pre-signing. We describe our model here informally and show a construction from adaptor signatures with rerandomizable keys.

An adaptor wallet considers one *master wallet*, which is used to deterministically initialize new wallets, so-called *child wallets.* Such child wallets are then used to generate (adaptor) signatures and are identified in our model by an identifier ID. In more detail, the master wallet generates and stores a master key pair $(\mathsf{msk}, \mathsf{mpk})$, a state St and a master statement/witness pair (Y_m, y_m) of a witness rerandomizable hard relation (cf. Def. 2). However, the master wallet is not used to generate signatures, but only to deterministically initialize child wallets, i.e., in order to initialize a child wallet with identifier ID, the master wallet deterministically derives a new key pair $(sk^{\mathsf{ID}}, pk^{\mathsf{ID}})$ from $(\mathsf{msk}, \mathsf{mpk})$, and a new statement/witness pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ from (Y_m, y_m). The child wallet can then use its key pair $(sk^{\mathsf{ID}}, pk^{\mathsf{ID}})$ to generate signatures and use its statement/witness pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ and a counter ctr to deterministically derive further statement/witness pairs. To keep our model simple, we do not allow child wallets to initialize further child wallets (as is done in the fully hierarchical setting [5]). We note, however, that our model can be extended to the fully hierarchical setting.

Similarly to the model of hierarchical deterministic wallets [5], we consider two kinds of child wallets, namely (1) non-hardened wallets, and (2) hardened wallets. Broadly speaking, the difference between these two is that we allow full corruption of hardened wallets, i.e., in our security games we allow the adversary to learn all secret values stored in a hardened wallet, including the session secret key sk^{ID}. For non-hardened wallets, on the other hand, we allow the adversary to only learn the session public key pk^{ID} and statement Y^{ID}. As a motivation for these two kinds of child wallets, recall the main applications of adaptor signatures as mentioned in the Introduction, namely payment channels and atomic swaps. A payment channel is typically used for frequent micropayments, i.e., users deposit only small amounts of money in a channel and use it often to sign transactions. In this case, it would be sensible to assume that the user operates the corresponding wallet on a mobile device, as it has to sign many transactions (possibly at a remote locations) and the impact of a wallet corruption is limited. Such a wallet would be represented by a hardened wallet in our model. On the other hand, atomic swaps are used, e.g., to swap coins of one cryptocurrency with coins of another currency. Such swaps are often one-time transactions of large amounts of funds or valuable tokens. In this example, it seems reasonable to implement the corresponding wallet as a hot/cold wallet, as it is crucial to secure such large amounts of funds or valuable tokens in the best possible way. The security goal for an adaptor wallet scheme is that the full corruption of hardened wallets does not compromise the security of any other (child or master) wallet. Additionally, we require that for all uncorrupted wallets, the derived public keys and statement/witness pairs are indistinguishable from freshly generated public keys and statement/witness pairs. Lastly, adaptor wallets must satisfy security notions similar to witness extractability under honestly rerandomizable keys (cf. Def. 11) and pre-signature adaptability (cf. Def. 9) of adaptor signatures with rerandomizable keys. Fig. 6 gives an illustration of our wallet model.

Statement/Witness rerandomization. According to the hot/cold wallet setting, it would be ideal if the deterministic derivation of statements and witnesses can be done independently. That is, we would like to store and derive statements exclusively on the hot wallet and witnesses only on the cold wallet. This would

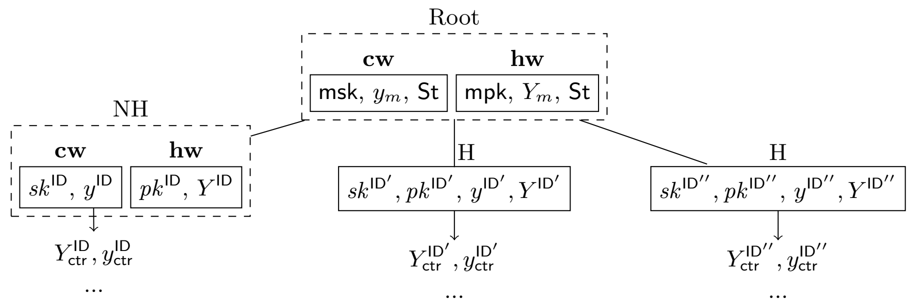

Root
cw
hw
msk, y_m, St
mpk, Y_m, St
NH
cw
hw
sk^{ID}, y^{ID}
pk^{ID}, Y^{ID}
H
$sk^{\mathsf{ID}'}, pk^{\mathsf{ID}'}, y^{\mathsf{ID}'}, Y^{\mathsf{ID}'}$
H
$sk^{\mathsf{ID}''}, pk^{\mathsf{ID}''}, y^{\mathsf{ID}''}, Y^{\mathsf{ID}''}$
$Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}}$
$Y^{\mathsf{ID}'}_{\mathsf{ctr}}, y^{\mathsf{ID}'}_{\mathsf{ctr}}$
$Y^{\mathsf{ID}''}_{\mathsf{ctr}}, y^{\mathsf{ID}''}_{\mathsf{ctr}}$

Fig. 6. Exemplatory design of our adaptor wallet scheme with three child wallets. H and NH denote hardened and non-hardened nodes respectively, **cw** and **hw** denote cold and hot wallets respectively and the values below the child wallets (e.g. $y^{\mathsf{ID}}_{\mathsf{ctr}}, Y^{\mathsf{ID}}_{\mathsf{ctr}}$) illustrate the statement/witness pairs that are being derived within each child wallet.

allow the cold wallet to stay entirely inactive (and therefore secure) in applications where it suffices to derive statements first and the corresponding witnesses only at a later time. Surprisingly, we show in Sec. 4.2 that for any multiplicative or additive statement/witness derivation, such an independent derivation is impossible. In our model and construction, we therefore resort to a joint statement/witness derivation.

An adaptor wallet scheme consists of a Setup algorithm, which initializes the master wallet, derivation algorithms for hardened and non-hardened keys ($\mathsf{SKDer_H}, \mathsf{PKDer_H}, \mathsf{SKDer_{NH}}, \mathsf{PKDer_{NH}}$) as well as for statement/witness pairs RDer, adaptor signature algorithms ($\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext}$) and signing and verification algorithms ($\mathsf{Sign}, \mathsf{Verify}$).

We now provide our generic construction of adaptor wallets, from an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma} = (\mathsf{RandSK}, \mathsf{RandPK}, \mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$. This construction uses a hash function $\mathsf{H} : \{0,1\}^* \to X$ and we require that the hard relation R is witness rerandomizable as per Def. 2. Our construction can be found in Figure 7.

4.2 Impossibility of Independent Statement/Witness Derivation

As mentioned above, one main question that arises when modeling derivation of statement/witness pairs in a deterministic fashion is whether an independent derivation of statement/witness pairs in hot and cold wallets respectively is possible. Surprisingly, unlike the secret and public key derivation mechanism, we show that this is not necessarily the case. At a high level, this is because unlike session secret keys, derived witnesses do not remain secret but are typically revealed in adaptor signature applications. More formally, we say that a hard relation $R \subseteq \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ has independently rerandomizable statement/witness pairs, if there exist two functions $f_{\mathsf{STDer}} : \mathcal{D}_{\mathsf{Y}} \times \{0,1\}^* \to \mathcal{D}_{\mathsf{Y}}$ and $f_{\mathsf{WitDer}} : \mathcal{D}_{\mathsf{w}} \times \{0,1\}^* \to \mathcal{D}_{\mathsf{w}}$ where for any $\rho \in \{0,1\}^*$ and any $(Y, y) \in R$ we have: $Y' \leftarrow f_{\mathsf{STDer}}(Y, \rho)$, $y' \leftarrow f_{\mathsf{WitDer}}(y, \rho)$, and $(Y', y') \in R$.

Translating the above to the hot/cold wallet setting, means that the cold wallet executes function f_{WitDer} and the hot wallet function f_{STDer}. An adversary in this setting can corrupt the hot wallet but not the cold wallet, and hence can learn the statements Y and Y' as well as the respective randomness ρ. In addition, as required by certain adaptor signature applications, the adversary eventually learns a derived witness $y' \leftarrow f_{\mathsf{WitDer}}(y, \rho)$. Therefore, if there exists a function $f^{-1} : \mathcal{D}_{\mathsf{w}} \times \{0,1\}^* \to \mathcal{D}_{\mathsf{w}}$ which on input y', ρ returns y, i.e., $y \leftarrow f^{-1}(y', \rho)$, then we cannot construct deterministic and independent statement/witness derivation from f_{WitDer} and f_{STDer}. This is, because an adversary could compute y and thereby break unforgeability of the adaptor wallet scheme. In the full version of this paper, we formalize this claim and prove it.

Let us now see how this result affects existing adaptor signature constructions. For the ECDSA-based adaptor signature construction $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ as described in Sec. 2.4 it is not possible to define f_{STDer} without providing the witness as input. This is mainly because the hard relation $R_g := \{((Y, \pi), y) \mid Y = g^y \wedge \mathsf{V}_g(Y, \pi) = 1\}$ requires a zero-knowledge proof alongside the statement Y, that proves knowledge of the witness y. Naturally, generating this proof without the witness is not possible. Now consider the "pure" dlog hard relation $R^{dlog} := \{(Y, y) \mid Y = g^y\}$, which is required for adaptor signature schemes based

```
Algorithm $\mathsf{Setup}(1^n)$
00 $\mathsf{St} \leftarrow_\$ \{0,1\}^n$
01 $(Y_m, y_m) \leftarrow R.\mathsf{GenR}(1^n)$
02 $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Gen}(1^n)$
03 Return $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m)$

Algorithm $\mathsf{pSign}(sk^{\mathsf{ID}}, m, Y)$
04 $\tilde{\sigma} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{pSign}(sk^{\mathsf{ID}}, m, Y)$
05 Return $\tilde{\sigma}$

Algorithm $\mathsf{pVrfy}(pk^{\mathsf{ID}}, m, Y, \tilde{\sigma})$
06 Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{pVrfy}(pk^{\mathsf{ID}}, m, Y, \tilde{\sigma})$

Algorithm $\mathsf{Adapt}(\tilde{\sigma}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
07 $\sigma \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Adapt}(\tilde{\sigma}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
08 Return $\sigma$

Algorithm $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathsf{ctr}})$
09 Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{Ext}(\sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathsf{ctr}})$

Algorithm $\mathsf{Sign}(sk^{\mathsf{ID}}, m)$
10 $\sigma \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Sign}(sk^{\mathsf{ID}}, m)$
11 Return $\sigma$

Algorithm $\mathsf{Verify}(pk^{\mathsf{ID}}, m, \sigma)$
12 Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{Verify}(pk^{\mathsf{ID}}, m, \sigma)$

Algorithm $\mathsf{SKDer_H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
13 $\rho \leftarrow \mathsf{H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
14 $sk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandSK}(\mathsf{msk}, \rho)$
15 Return $sk^{\mathsf{ID}}$

Algorithm $\mathsf{SKDer_{NH}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
16 $\rho \leftarrow \mathsf{H}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
17 $sk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandSK}(\mathsf{msk}, \rho)$
18 Return $sk^{\mathsf{ID}}$

Algorithm $\mathsf{PKDer_H}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
19 $\rho \leftarrow \mathsf{H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
20 $pk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho)$
21 Return $pk^{\mathsf{ID}}$

Algorithm $\mathsf{PKDer_{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
22 $\rho \leftarrow \mathsf{H}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
23 $pk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho)$
24 Return $pk^{\mathsf{ID}}$

Algorithm $\mathsf{RDer}(Y, y, \mathsf{ctr}, \mathsf{ID})$
25 $\rho \leftarrow \mathsf{H}(y, \mathsf{ctr}, \mathsf{ID})$
26 $y^{\mathsf{ID}}_{\mathsf{ctr}} \leftarrow R.\mathsf{RandWit}(y, \rho)$
27 $Y^{\mathsf{ID}}_{\mathsf{ctr}} \leftarrow R.\mathsf{WitToSt}(y^{\mathsf{ID}}_{\mathsf{ctr}})$
28 Return $(Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
```

Fig. 7. Generic construction of adaptor wallets w.r.t. an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$, where R is a witness rerandomizable hard relation as per Def. 2 and a hash function $\mathsf{H} : \{0,1\}^* \to X$.

on Schnorr and Katz-Wang [8]. The statement/witness pairs for this relation can be rerandomized either multiplicatively or additively. Both of these operations, however, can easily be inverted. For instance, for a statement/witness pair $(g^y, y) \in R^{dlog}$, an additive rerandomization would instantiate the functions f_{STDer} and f_{WitDer} as $f_{\mathsf{STDer}}(g^y, \rho) := g^y \cdot g^\rho = Y'$ and $f_{\mathsf{WitDer}}(y, \rho) := y + \rho = y'$. Naturally, the function f^{-1} can simply be instantiated as $f^{-1}(y', \rho) := y' - \rho = y$.

Impact of the impossibility result. Due to the above impossibility result of independent statement/witness derivation we cannot construct an adaptor wallet scheme with statement derivation in the hot wallet. However, for certain applications of adaptor signatures, this restriction is tolerable as the cold wallet does not need to generate many signatures and/or statement/witness pairs and therefore does not need to be activated frequently. Further, in practice one can minimize the number of times a cold wallet must be activated by batching the generation of statement/witness pairs, i.e., the cold wallet can generate multiple pairs and send all statements at once to the hot wallet. For other applications with frequent transactions, such as payment channels, an adaptor wallet user can use a hardened wallet as explained in Sec. 4.1.

Acknowledgments. This work is supported by the German Research Foundation DFG - SFB 1119 - 236615297 (CROSSING Project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain Project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE.*

References

[1] N. Alkeilani Alkadri et al. "Deterministic Wallets in a Quantum World". In: *ACM CCS 2020.* Ed. by J. Ligatti, X. Ou, J. Katz, and G. Vigna. ACM Press, Nov. 2020, pp. 1017–1031. DOI: 10.1145/3372297.3423361.

[2] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken. “Dynamic and Efficient Key Management for Access Hierarchies”. In: *ACM Trans. Inf. Syst. Secur.* 12.3 (Jan. 2009). ISSN: 1094-9224. DOI: 10.1145/1455526.1455531. URL: https://doi.org/10.1145/1455526.1455531.

[3] L. Aumayr et al. “Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures”. In: *Advances in Cryptology – ASIACRYPT 2021*. Ed. by M. Tibouchi and H. Wang. Cham: Springer International Publishing, 2021, pp. 635–664.

[4] M. Blum, P. Feldman, and S. Micali. “Non-Interactive Zero-Knowledge and Its Applications”. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC ’88. Chicago, Illinois, USA: Association for Computing Machinery, 1988, 103–112. ISBN: 0897912640. DOI: 10.1145/62212.62222. URL: https://doi.org/10.1145/62212.62222.

[5] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi. “The Exact Security of BIP32 Wallets”. In: *ACM CCS 2021*. Ed. by G. Vigna and E. Shi. ACM Press, Nov. 2021, pp. 1020–1042. DOI: 10.1145/3460120.3484807.

[6] P. Das, S. Faust, and J. Loss. “A Formal Treatment of Deterministic Wallets”. In: *ACM CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM Press, Nov. 2019, pp. 651–668. DOI: 10.1145/3319535.3354236.

[7] A. Deshpande and M. Herlihy. “Privacy-Preserving Cross-Chain Atomic Swaps”. In: *FC 2020*. Ed. by M. Bernhard et al. Springer International Publishing, 2020.

[8] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. “Two-Party Adaptor Signatures from Identification Schemes”. In: *PKC 2021, Part I*. Ed. by J. Garay. Vol. 12710. LNCS. Springer, Heidelberg, May 2021, pp. 451–480. DOI: 10.1007/978-3-030-75245-3_17.

[9] M. F. Esgin, O. Ersoy, and Z. Erkin. “Post-Quantum Adaptor Signatures and Payment Channel Networks”. In: *ESORICS 2020, Part II*. Ed. by L. Chen, N. Li, K. Liang, and S. A. Schneider. Vol. 12309. LNCS. Springer, Heidelberg, Sept. 2020, pp. 378–397. DOI: 10.1007/978-3-030-59013-0_19.

[10] M. Fersch, E. Kiltz, and B. Poettering. “On the One-Per-Message Unforgeability of (EC)DSA and Its Variants”. In: *TCC 2017, Part II*. Ed. by Y. Kalai and L. Reyzin. Vol. 10678. LNCS. Springer, Heidelberg, Nov. 2017, pp. 519–534. DOI: 10.1007/978-3-319-70503-3_17.

[11] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. “Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys”. In: *PKC 2016, Part I*. Ed. by C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang. Vol. 9614. LNCS. Springer, Heidelberg, Mar. 2016, pp. 301–330. DOI: 10.1007/978-3-662-49384-7_12.

[12] G. Gutoski and D. Stebila. “Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage”. In: *FC 2015*. Ed. by R. Böhme and T. Okamoto. Vol. 8975. LNCS. Springer, Heidelberg, Jan. 2015, pp. 497–504. DOI: 10.1007/978-3-662-47854-7_31.

[13] J. Katz and N. Wang. “Efficiency Improvements for Signature Schemes with Tight Security Reductions”. In: *ACM CCS 2003*. Ed. by S. Jajodia, V. Atluri, and T. Jaeger. ACM Press, Oct. 2003, pp. 155–164. DOI: 10.1145/948109.948132.

[14] Y. Kondi, B. Magri, C. Orlandi, and O. Shlomovits. “Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 608–625. DOI: 10.1109/SP40001.2021.00067.

[15] A. D. Luzio, D. Francati, and G. Ateniese. “Arcula: A Secure Hierarchical Deterministic Wallet for Multi-asset Blockchains”. In: *CANS 20*. Ed. by S. Krenn, H. Shulman, and S. Vaudenay. Vol. 12579. LNCS. Springer, Heidelberg, Dec. 2020, pp. 323–343. DOI: 10.1007/978-3-030-65411-5_16.

[16] V. Madathil, S. A. Thyagarajan, D. Vasilopoulos, L. Fournier, G. Malavolta, and P. Moreno-Sanchez. *Practical Decentralized Oracle Contracts for Cryptocurrencies*. Cryptology ePrint Archive, Report 2022/499. https://ia.cr/2022/499. 2022.

[17] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei. “Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability”. In: *NDSS 2019*. The Internet Society, Feb. 2019.

[18] P. Moreno-Sanchez and A. Kate. *Scriptless Scripts with ECDSA*. https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf. 2018.

[19] A. Poelstra. *Scriptless scripts.* `https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf`. 2017. Visited 10/2020.

[20] E. B. Sasson et al. "Zerocash: Decentralized anonymous payments from bitcoin". In: *2014 IEEE symposium on security and privacy.* IEEE. 2014, pp. 459–474.

[21] C.-P. Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: `10.1007/BF00196725`.

[22] E. Tairi, P. Moreno-Sanchez, and M. Maffei. "A^2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs". In: *2021 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, May 2021, pp. 1834–1851. DOI: `10.1109/SP40001.2021.00111`.

[23] E. Tairi, P. Moreno-Sanchez, and M. Maffei. "Post-Quantum Adaptor Signature for Privacy-Preserving Off-Chain Payments". In: *Financial Cryptography and Data Security.* Ed. by N. Borisov and C. Diaz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 131–150. ISBN: 978-3-662-64331-0.

[24] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sánchez. *Universal Atomic Swaps: Secure Exchange of Coins Across All Blockchains.* Cryptology ePrint Archive, Report 2021/1612. `https://ia.cr/2021/1612`. 2021.

[25] N. Van Saberhagen. "CryptoNote v 2.0". In: (2013).

[26] X. Yin, Z. Liu, G. Yang, G. Chen, and H. Zhu. *Secure Hierarchical Deterministic Wallet Supporting Stealth Address.* Cryptology ePrint Archive, Paper 2022/627. `https://eprint.iacr.org/2022/627`. 2022. URL: `https://eprint.iacr.org/2022/627`.

E. Two-Party Adaptor Signatures from Identification Schemes

In this chapter, we present an adjusted version of the following publication

[77] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. "Two-Party Adaptor Signatures from Identification Schemes". In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I.* 2021, pp. 451–480. **Part of this thesis.**

Concretely, the work included in this chapter differs from the above publication [77] in the following ways: (1) Theorems 3 and 4 have been adjusted such that they explicitly state that NIZK is a non-interactive zero-knowledge proof system and C is an extractable commitment scheme, and (2) some typos have been fixed in the proof of Lemma 6 and the overall write-up of the proof has been improved.

Two-Party Adaptor Signatures From Identification Schemes

Andreas Erwig[1], Sebastian Faust[1], Kristina Hostáková[2,†], Monosij Maitra[1,‡], and Siavash Riahi[1]

[1] Technische Universität Darmstadt, Germany
firstname.lastname@tu-darmstadt.de
[2] ETH Zürich, Switzerland
kristina.hostakova@inf.ethz.ch

Abstract. Adaptor signatures are a novel cryptographic primitive with important applications for cryptocurrencies. They have been used to construct second layer solutions such as payment channels or cross-currency swaps. The basic idea of an adaptor signature scheme is to tie the signing process to the revelation of a secret value in the sense that, much like a regular signature scheme, an adaptor signature scheme can authenticate messages, but simultaneously leaks a secret to certain parties. Recently, Aumayr et al. provide the first formalization of adaptor signature schemes, and present provably secure constructions from ECDSA and Schnorr signatures. Unfortunately, the formalization and constructions given in this work have two limitations: (1) current schemes are limited to ECDSA and Schnorr signatures, and no generic transformation for constructing adaptor signatures is known; (2) they do not offer support for aggregated two-party signing, which can significantly reduce the blockchain footprint in applications of adaptor signatures.

In this work, we address these two shortcomings. First, we show that signature schemes that are constructed from identification (ID) schemes, which additionally satisfy certain homomorphic properties, can generically be transformed into adaptor signature schemes. We further provide an impossibility result which proves that unique signature schemes (e.g., the BLS scheme) cannot be transformed into an adaptor signature scheme. In addition, we define two-party adaptor signature schemes with aggregatable public keys and show how to instantiate them via a generic transformation from ID-based signature schemes. Finally, we give instantiations of our generic transformations for the Schnorr, Katz-Wang and Guillou-Quisquater signature schemes.

1 Introduction

Blockchain technologies, envisioned first in 2009 [33], have spurred enormous interest by academia and industry. This technology puts forth a decentralized payment paradigm, where financial transactions are stored in a decentralized data structure – often referred to as the blockchain. The main cryptographic primitive used by blockchain systems is the one of digital signature schemes, which allow users to authenticate payment transactions. Various different flavors of digital signature schemes are used by blockchain systems, e.g., ring signatures [38] add privacy-preserving features to cryptocurrencies [39], while threshold signatures and multi-signatures are used for multi-factor authorization of transactions [18].

Adaptor signatures (sometimes also referred to as scriptless scripts) are another important type of digital signature scheme introduced by the cryptocurrency community [36] and recently formalized by Aumayr et al. [2]. In a nutshell, adaptor signatures tie together authorization of a message and the leakage of a secret value. Namely, they allow a *signer* to produce a *pre-signature* under her secret key such that this pre-signature can be *adapted* into a valid signature by a *publisher* knowing a certain secret value. If the completed signature gets published, the signer is able to extract the embedded secret used by the publisher.

To demonstrate the concept of adaptor signatures, let us discuss the simple example of a preimage sale which serves as an important building block in many blockchain applications such as payment channels [6, 10,

† Research partially conducted at Technische Universität Darmstadt, Germany.
‡ Research partially conducted at Indian Institute of Technology Madras, India.

37, 2], payment routing in payment channel networks (PCNs) [29, 13, 32] or atomic swaps [11, 21]. Assume that a seller offers to reveal a preimage of a hash value h in exchange for c coins from a concrete buyer. This is a classical instance of a fair exchange problem, which can be solved using the blockchain as follows. The buyer locks c coins in a transaction which can be spent by another transaction if it is authorized by the seller and contains a preimage of the hash value h.

While this solution implements the preimage sale, it has various drawbacks: (i) The only hash functions that can be used are the ones supported by the underlying blockchain. For example, the most popular blockchain-based cryptocurrency, Bitcoin, supports only SHA-1, SHA-256 and RIPEMD-160 [5]. This makes the above solution unsuitable for applications like privacy-preserving payment routing in PCNs [29, 13] that crucially rely on the preimage sale instantiated with a *homomorphic* hash function. (ii) The hash value has to be fixed at the beginning of the sale and cannot be changed later without a new transaction being posted on the blockchain. This is problematic in, e.g., generalized payment channels [2], where users utilize the ideas from the preimage sale to repeatedly update channel balances without any blockchain interaction. (iii) Finally, the blockchain script is non-standard as, in addition to a signature verification, it contains a hash preimage verification. This does not only make the transaction more expensive but also allows parties who are maintaining the blockchain (also known as *miners*) to censor transactions belonging to a preimage sale.

The concept of adaptor signatures allows us to implement a preimage sale in a way that overcomes most of the aforementioned drawbacks. The protocol works at a high level as follows. The buyer locks c coins in a transaction which can be spent by a transaction authorized by *both* the seller and the buyer. Thereafter, the buyer pre-signs a transaction spending the c coins with respect to the hash value h. If the seller knows a preimage of h, she can adapt the pre-signature of the buyer, attach her own signature and claim the c coins. The buyer can then extract a preimage from the adapted signature. Hence, parties are not restricted to the hash functions supported by the blockchain, i.e., drawback (i) is addressed. Moreover, the buyer can pre-sign the spending transaction with respect to multiple hash values which overcomes drawback (ii). However, the third drawback remains. While the usage of adaptor signatures avoids the hash preimage verification in the script, it adds a signature verification (i.e., there are now 2 signature verifications in total) which makes this type of exchange easily distinguishable from a normal payment transaction. Hence, the sale remains rather expensive and censorship is not prevented.

The idea of *two-party* adaptor signatures is to replace the two signature verifications by one. The transaction implementing a preimage sale then has exactly the same format as a transaction simply transferring coins. As a result the price (in terms of fees paid to the miners) of the preimage sale transaction is the same as the price for a normal payment. Moreover, censorship is prevented as miners cannot distinguish the transactions belonging to the preimage sale from a standard payment transaction. Hence, point (iii) is fully addressed.

The idea of replacing two signatures by one has already appeared in the literature in the context of payment channels. Namely, Malavolta et al. [29] presented protocols for two-party threshold adaptor signatures based on Schnorr and ECDSA digital signatures. However, they did not present a standalone definition for the threshold primitive and hence security for these schemes has not been analyzed. Furthermore, the key generation of the existing threshold adaptor signature schemes is interactive which is undesirable. Last but not least, their constructions are tailored to Schnorr and ECDSA signature schemes and hence is not generic. From the above points, the following natural question arises:

Is it possible to define and instantiate two-party adaptor signature schemes with non-interactive key generation in a generic way?

1.1 Our contribution

Our main goal is to define two-party adaptor signatures and explore from which digital signature we can instantiate this new primitive. We proceed in three steps which we summarize below and depict in Fig. 1.

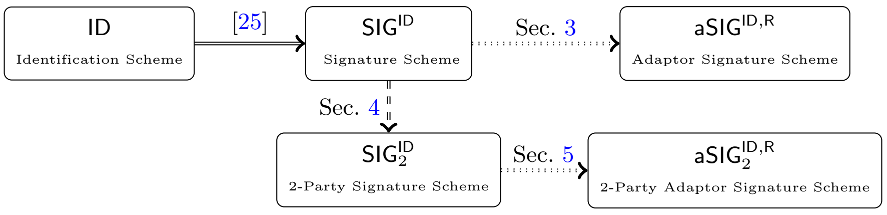
Step 1: From ID schemes to adaptor signatures. Our first goal is to determine if there exists a specific class of signature schemes which can be generically transformed into adaptor signatures. Given the existing

Schnorr-based construction [36, 2], a natural choice is to explore signature schemes constructed in a similar fashion. To this end, we focus on signature schemes built from identification (ID) schemes using the Fiat-Shamir transform [25]. We show that ID-based signature schemes satisfying certain additional properties can be transformed to adaptor signature schemes generically. In addition to Schnorr signatures [40], this class includes Katz-Wang and Guillou-Quisquater signatures [24, 22]. As an additional result, we show that adaptor signatures *cannot* be built from unique signatures, ruling out constructions from, e.g., BLS signatures [9].

Our generic transformation of adaptor signatures from ID schemes has multiple benefits. Firstly, by instantiating it with the Guillou-Quisquater siganture scheme, we obtain the first RSA-based adaptor signature scheme. Secondly, since Katz-Wang signatures offers tight security (under the decisional Diffie-Hellman (DDH) assumption), and our generic transformation also achieves tight security, our result shows how to construct adaptor signatures with a tight reduction to the underlying DDH assumption.

Step 2: From ID schemes to two-party signatures. Our second goal is to generically transform signature schemes built from ID schemes into two-party signature schemes with aggregatable public keys. Unlike threshold signatures, these signatures have non-interactive key generation. This means that parties can independently generate their key pairs and later collaboratively generate signatures that are valid under their *combined* public key. For our transformation, we require the signature scheme to satisfy certain aggregation properties which, as we show, are present in the three aforementioned signature schemes. While this transformation serves as a middle step towards our main goal of constructing two-party adaptor signatures, we believe it is of independent interest.

Step 3: From ID schemes to two-party adaptor signatures. Finally, we define two-party adaptor signature schemes with aggregatable public keys. In order to instantiate this novel cryptographic primitive, we use similar techniques as in step 1 where we "lifted" standard signature schemes to adaptor signature schemes. More precisely, we present a transformation turning a two-party signature scheme based on an ID scheme into a two-party adaptor signature scheme.



ID
Identification Scheme
[25]
$\mathsf{SIG}^{\mathsf{ID}}$
Signature Scheme
Sec. 3
$\mathsf{aSIG}^{\mathsf{ID},R}$
Adaptor Signature Scheme
Sec. 4
$\mathsf{SIG}_2^{\mathsf{ID}}$
2-Party Signature Scheme
Sec. 5
$\mathsf{aSIG}_2^{\mathsf{ID},R}$
2-Party Adaptor Signature Scheme

Fig. 1: Overview of our results. Full arrow represents a generic transformation, dotted and dashed arrows represent a generic transformation which requires additional homomorphic or aggregation properties respectively.

Remark 1. Let us point out that Fig. 1 presents our transformation steps from signature schemes based on ID schemes to two-party adaptor signatures. Despite the fact that we generically construct our two-party adaptor signature scheme from two-party signature schemes based on ID schemes, we reduce its security to the strong unforgeability of the underlying single party signature scheme. Therefore, we do not need the two-party signature scheme from ID schemes to be strongly unforgeable. This gives us a more general result than proving security based on strong unforgeability of the two-party signature scheme from ID schemes. We note that any ID scheme can be transformed to a signature scheme with strong unforgeability by Bellare and Shoup [4].

Let us further mention that our security proofs are in the random oracle model. Proving the security of our constructions and the original constructions from [2] in the standard model remains an interesting open problem.

1.2 Related Work

Adaptor Signatures. The notion of adaptor signatures was first introduced by Poelstra [36] and has since been used in many blockchain related applications, such as PCNs [29], payment channel hubs [42] or atomic swaps [11]. However, the adaptor signatures as a standalone primitive were only formalized later by Aumayr et al. [2], where they were used to generalize the concept of payment channels. Concurrently, Fournier [17] attempted to formalize adaptor signatures, however, as pointed out in [2], his definition is weaker than the one given in [2] and not sufficient for certain applications. All the previously mentioned works constructed adaptor signatures only from Schnorr and ECDSA signatures, i.e., they did not show generic transformations for building adaptor signature schemes. As previously mentioned, a two-party threshold variant of adaptor signatures was presented by Malavolta et al. [29]. Their construction requires interactive key generation, thereby differing from our two-party adaptor signature notion. Moreover, no standalone definition of the threshold primitive was provided.

Two works [15, 43] have recently introduced post-quantum secure adaptor signature schemes, i.e., schemes that remain secure even in presence of an adversary having access to a quantum computer. In order to achieve post-quantum security, [15] based its scheme on standard and well-studied lattice assumptions, namely Module-SIS and Module-LWE, while the scheme in [43] is based on lesser known assumptions for isogenies. Both works additionally show how to construct post-quantum secure PCNs from their respective adaptor signature schemes.

Multi-Signatures and ID Schemes. Multi-Signatures have been subject to extensive research in the past (e.g., [35, 34, 23]). In a nutshell, multi-signatures allow a set of signers to collaboratively generate a signature for a common message such that the signature can be verified given the public key of each signer. More recently, the notion of multi-signatures with aggregatable public keys has been introduced [30] and worked on [8, 26], which allows to aggregate the public keys of all signers into one single public key. We use some results from the work of Kiltz et al. [25], which provides a concrete and modular security analysis of signatures schemes from ID schemes obtained via the Fiat-Shamir transformation. Our paper builds up on their work and uses some of their notation.

2 Preliminaries

In this section, we introduce notation that we use throughout this work and preliminaries on adaptor signatures and identification schemes. Due to space limitations, we provide formal definitions of digital signature schemes, non-interactive zero-knowledge proofs and extractable commitments in the full version of this paper [14].

Notation. We denote by $x \leftarrow_{\$} \mathcal{X}$ the uniform sampling of x from the set $\mathcal{X}$. Throughout this paper, n denotes the security parameter. By $x \leftarrow \mathsf{A}(y)$ we denote a *probabilistic polynomial time* (PPT) algorithm A that on input y, outputs x. When A is a *deterministic polynomial time* (DPT) algorithm, we use the notation $x := \mathsf{A}(y)$. A function $\nu\colon \mathbb{N} \to \mathbb{R}$ is *negligible in* n if for every $k \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ s.t. for every $n \geq n_0$ it holds that $|\nu(n)| \leq 1/n^k$.

Hard relation. Let $\mathsf{R} \subseteq \mathcal{D}_S \times \mathcal{D}_{\mathsf{w}}$ be a relation with statement/witness pairs $(Y, y) \in \mathcal{D}_S \times \mathcal{D}_{\mathsf{w}}$ and let the language $L_{\mathsf{R}} \subseteq \mathcal{D}_S$ associated to R be defined as $L_{\mathsf{R}} := \{Y \in \mathcal{D}_S \mid \exists y \in \mathcal{D}_{\mathsf{w}} \text{ s.t. } (Y, y) \in \mathsf{R}\}$. We say that R is a *hard relation* if: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^n)$ that on input the security parameter outputs a pair $(Y, y) \in \mathsf{R}$; (ii) The relation R is poly-time decidable; (iii) For all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a valid witness $y \in \mathcal{D}_{\mathsf{w}}$ for $Y \in L_{\mathsf{R}}$ is negligible.

2.1 Adaptor Signatures

We now recall the definition of adaptor signatures, recently put forward in [2].

Definition 1 (Adaptor signature). *An adaptor signature scheme w.r.t. a hard relation* R *and a signature scheme* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of a tuple of four algorithms* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ *defined as:*

$\mathsf{pSign}_{sk}(m, Y)$**:** *is a PPT algorithm that on input a secret key sk, message* $m \in \{0,1\}^*$ *and statement* $Y \in L_\mathsf{R}$*, outputs a pre-signature* $\widetilde{\sigma}$*.*

$\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma})$**:** *is a DPT algorithm that on input a public key pk, message* $m \in \{0,1\}^*$*, statement* $Y \in L_\mathsf{R}$ *and pre-signature* $\widetilde{\sigma}$*, outputs a bit b.*

$\mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$**:** *is a DPT algorithm that on input a pre-signature* $\widetilde{\sigma}$ *and witness y, outputs a signature* σ*.*

$\mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$**:** *is a DPT algorithm that on input a signature* σ*, pre-signature* $\widetilde{\sigma}$ *and statement* $Y \in L_\mathsf{R}$*, outputs a witness y such that* $(Y, y) \in \mathsf{R}$*, or* $\bot$*.*

An adaptor signature scheme, besides satisfying plain digital signature correctness, should also satisfy pre-signature correctness that we formalize next.

Definition 2 (Pre-signature correctness). *An adaptor signature* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ *satisfies* pre-signature correctness*, if for all* $n \in \mathbb{N}$ *and* $m \in \{0,1\}^*$*:*

$$\Pr\left[\begin{array}{c|l} \mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1 \wedge & (sk, pk) \leftarrow \mathsf{Gen}(1^n), (Y, y) \leftarrow \mathsf{GenR}(1^n) \\ \mathsf{Vrfy}_{pk}(m; \sigma) = 1 \wedge & \widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y), \sigma := \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y) \\ (Y, y') \in \mathsf{R} & y' := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y) \end{array}\right] = 1.$$

An adaptor signature scheme $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ is called *secure* if it satisfies three security properties: *existential unforgeablity under chosen message attack for adaptor signatures*, *pre-signature adaptability* and *witness extractability.* Let us recall the formal definition of these properties next.

The notion of unforgeability for adaptor signatures is similar to existential unforgeability under chosen message attacks for standard digital signatures but additionally requires that producing a forgery σ for some message m^* is hard even given a pre-signature on m^* w.r.t. a random statement $Y \in L_\mathsf{R}$.

Definition 3 ($\mathsf{aEUF{-}CMA}$ Security). *An adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ *is unforgeable if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n) = 1] \leq \nu(n)$*, where the definition of the experiment* $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}$ *is as follows:*

```
$\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n)$
  1 : $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
  2 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(pk)$
  3 : $(Y, y) \leftarrow \mathsf{GenR}(1^n), \widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y)$
  4 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(\widetilde{\sigma}, Y)$
  5 : return $\big(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*)\big)$

$\mathcal{O}_\mathrm{S}(m)$
  1 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
  2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  3 : return $\sigma$

$\mathcal{O}_\mathrm{pS}(m, Y)$
  1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
  2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  3 : return $\widetilde{\sigma}$
```

A natural requirement for an adaptor signature scheme is that any valid pre-signature w.r.t. Y (possibly produced by a malicious signer) can be completed into a valid signature using a witness y with $(Y, y) \in \mathsf{R}$.

Definition 4 (Pre-signature adaptability). *An adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{SIG},\mathsf{R}}$ *satisfies pre-signature adaptability, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, statement/witness pairs* $(Y, y) \in \mathsf{R}$*, public keys pk and pre-signatures* $\widetilde{\sigma} \leftarrow \{0,1\}^*$ *we have* $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$*, then* $\mathsf{Vrfy}_{pk}(m; \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)) = 1$*.*

The last property that we are interested in is *witness extractability.* Informally, it guarantees that a valid signature/pre-signatue pair $(\sigma, \widetilde{\sigma})$ for message/statement (m, Y) can be used to extract a corresponding witness y.

Definition 5 (Witness extractability). *An adaptor signature scheme* $\mathsf{aSIG_R}$ *is* witness extractable *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* ν *such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG_{R,SIG}}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG_{R,SIG}}}$ *is defined as follows:*

```
$\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG_{R,SIG}}}(n)$
  1 : $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
  2 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S},\mathcal{O}_\mathrm{pS}}(pk)$
  3 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$
  4 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S},\mathcal{O}_\mathrm{pS}}(\widetilde{\sigma})$
  5 : $y := \mathsf{Ext}_{pk}(\sigma^*, \widetilde{\sigma}, Y^*)$
  6 : return $(m^* \notin \mathcal{Q} \wedge (Y^*, y) \notin \mathsf{R} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*))$

$\mathcal{O}_\mathrm{S}(m)$
  1 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
  2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  3 : return $\sigma$

$\mathcal{O}_\mathrm{pS}(m, Y)$
  1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
  2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  3 : return $\widetilde{\sigma}$
```

Let us stress that while the witness extractability experiment $\mathsf{aWitExt}$ looks fairly similar to the experiment $\mathsf{aSigForge}$, there is one crucial difference; namely, the adversary is allowed to choose the forgery statement Y^*. Hence, we can assume that it knows a witness for Y^* and can thus generate a valid signature on the forgery message m^*. However, this is not sufficient to win the experiment. The adversary wins *only* if the valid signature does not reveal a witness for Y^*.

2.2 Identification and Signature Schemes

In this section we recall the definition of identification schemes and how they are transformed to signature schemes as described in [25].

Definition 6 (Canonical Identification Scheme [25]). *A canonical identification scheme* ID *is defined as a tuple of four algorithms* $\mathsf{ID} := (\mathsf{IGen}, \mathsf{P}, \mathsf{ChSet}, \mathsf{V})$.

– *The key generation algorithm* IGen *takes the system parameters* par *as input and returns secret and public key* (sk, pk)*. We assume that* pk *defines the set of challenges, namely* ChSet.
– *The prover algorithm* P *consists of two algorithms namely* P_1 *and* P_2*:*
  - P_1 *takes as input the secret key* sk *and returns a commitment* $R \in \mathcal{D}_\mathsf{rand}$ *and a state* St.
  - P_2 *takes as input the secret key* sk*, a commitment* $R \in \mathcal{D}_\mathsf{rand}$*, a challenge* $h \in \mathsf{ChSet}$*, and a state* St *and returns a response* $s \in \mathcal{D}_\mathsf{resp}$.
– *The verifier algorithm* V *is a deterministic algorithm that takes the public key* pk *and the conversation transcript as input and outputs* 1 *(acceptance) or* 0 *(rejection).*

We require that for all $(sk, pk) \in \mathsf{IGen}(\mathsf{par})$*, all* $(R, St) \in \mathsf{P}_1(sk)$*, all* $h \in \mathsf{ChSet}$ *and all* $s \in \mathsf{P}_2(sk, R, h, St)$*, we have* $\mathsf{V}(pk, R, h, s) = 1$.

We recall that an identification scheme ID is called *commitment-recoverable*, if V first internally calls a function V_0 which recomputes $R_0 = \mathsf{V}_0(pk, h, s)$ and then outputs 1, iff $R_0 = R$. Using Fiat-Shamir heuristic one can transform any identification scheme ID of the above form into a digital signature scheme $\mathsf{SIG^{ID}}$. We recall this transformation in Fig. 2 when ID is commitment-recoverable.

3 Adaptor Signatures from $\mathsf{SIG^{ID}}$

Our first goal is to explore and find digital signature schemes which can generically be transformed to adaptor signatures. Interestingly, we observe that both existing adaptor signature schemes, namely the Schnorr-based and the ECDSA-based schemes, utilize the randomness used during signature generation to transform digital signatures to adaptor signatures [2]. We first prove a negative result, namely that it is impossible to construct an adaptor signature scheme from a unique signature scheme [41, 28, 19]. Thereafter, we focus on signature schemes constructed from identification schemes (cf. Fig. 2) and show that if the underlying ID-based signature scheme $\mathsf{SIG^{ID}}$ satisfies certain additional properties, then we can generically transform it into an adaptor signature scheme. To demonstrate the applicability of our generic transformation, we show in the full version of this paper [14] that many existing $\mathsf{SIG^{ID}}$ instantiations satisfy the required properties.

```
$\mathsf{Gen}(1^n)$
    1 : $(sk, pk) \leftarrow \mathsf{IGen}(n)$
    2 : **return** $(sk, pk)$

$\mathsf{Sign}_{sk}(m)$
    1 : $(R, St) \leftarrow \mathsf{P}_1(sk)$
    2 : $h := \mathcal{H}(R, m)$
    3 : $s \leftarrow \mathsf{P}_2(sk, R, h, St)$
    4 : **return** $(h, s)$

$\mathsf{Vrfy}_{pk}(m; (h, s))$
    1 : $R := \mathsf{V}_0(pk, h, s)$
    2 : **return** $h = \mathcal{H}(R, m)$
```

Fig. 2: $\mathsf{SIG}^{\mathsf{ID}}$: Digital signature schemes from identification schemes [25].

3.1 Impossibility Result for Unique Signatures

An important class of digital signatures are those where the signing algorithm is deterministic and the generated signatures are unique. Given the efficiency of deterministic signature schemes along with numerous other advantages that come from signatures being unique [41, 28, 19], it would be tempting to design adaptor signatures based on unique signatures. However, we show in Thm. 1 that if the signature scheme has unique signatures, then it is impossible to construct a secure adaptor signature scheme from it.

Theorem 1. *Let* R *be a hard relation and* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a signature scheme with unique signatures. Then there does not exist an adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$.

Proof. We prove this theorem by contradiction. Assume there exists an adaptor signature scheme where the underlying signature scheme, SIG, has unique signatures. We construct a PPT algorithm $\mathcal{A}$ which internally uses the adaptor signature and breaks the hardness of R. In other words, $\mathcal{A}$ receives $(1^n, Y)$ as input and outputs y, such that $(Y, y) \in \mathsf{R}$. Below, we describe $\mathcal{A}$ formally.

```
On input $(1^n, Y)$, $\mathcal{A}$ proceeds as follows:
    1 : Sample a new key pair $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$.
    2 : Choose an arbitrary message $m$ from the signing message space.
    3 : Generate a pre-signature, $\widetilde{\sigma} \leftarrow \mathsf{preSign}_{sk}(m, Y)$.
    4 : Generate a signature, $\sigma := \mathsf{Sign}_{sk}(m)$.
    5 : Compute and output $y := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$.
```

We now show that y returned by $\mathcal{A}$ is indeed a witness of Y, i.e., $(Y, y) \in \mathsf{R}$. From the correctness of the adaptor signature scheme, we know that for any y' s.t. $(Y, y') \in \mathsf{R}$ the signature $\sigma' := \mathsf{Adapt}(\widetilde{\sigma}, y')$ is a valid signature, i.e., $\mathsf{Vrfy}_{pk}(m, \sigma') = 1$. Moreover, we know that $y'' := \mathsf{Ext}_{pk}(\sigma', \widetilde{\sigma}, Y)$ is such that $(Y, y'') \in \mathsf{R}$. As SIG is a unique signature scheme, this implies that $\sigma' = \sigma$ which in turn implies that the witness y returned by $\mathcal{A}$ is y''. Hence, $\mathcal{A}$ breaks the hardness of R with probability 1.

Let us briefly discuss which signature schemes are affected by our impossibility result. Unique signature schemes (also known as verifiable unpredictable functions (VUF)) have been first introduced in [19]. Furthermore, many follow-up works such as [31, 28] and most recently [41], have shown how to instantiate this primitive in the standard model. Another famous example of a unique signature scheme is BLS [9]. Naturally, due to our impossibility result, an adaptor signature scheme cannot be instantiated from these signature schemes.

3.2 Generic Transformation to Adaptor Signatures

We now describe how to generically transform a randomized digital signature scheme $\mathsf{SIG}^{\mathsf{ID}}$ from Fig. 2 into an adaptor signature scheme w.r.t. a hard relation R. For brevity, we denote the resulting adaptor signature scheme as $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ instead of $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}^{\mathsf{ID}}}$. The main idea behind our transformation is to *shift* the public randomness of the Sign procedure by a statement Y for the relation R in order to generate a modified signature called a *pre-signature*. Using a corresponding witness y (i.e., $(Y, y) \in \mathsf{R}$), the shift of the public randomness in the pre-signature can be reversed (or adapted), in order to obtain a regular (or full) signature. Moreover, it should be possible to extract a witness given both the pre-signature and the full-signature. To this end, let us formalize three new *deterministic* functions which we will use later in our transformation.

1. For the randomness shift, we define a function $f_{\mathsf{shift}} \colon \mathcal{D}_{\mathsf{rand}} \times L_{\mathsf{R}} \to \mathcal{D}_{\mathsf{rand}}$ that takes as input a commitment value $R \in \mathcal{D}_{\mathsf{rand}}$ of the identification scheme and a statement $Y \in L_{\mathsf{R}}$ of the hard relation, and outputs a new commitment value $R' \in \mathcal{D}_{\mathsf{rand}}$.
2. For the adapt operation, we define $f_{\mathsf{adapt}} \colon \mathcal{D}_{\mathsf{resp}} \times \mathcal{D}_{\mathsf{w}} \to \mathcal{D}_{\mathsf{resp}}$ that takes as input a response value $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$ of the identification scheme and a witness $y \in \mathcal{D}_{\mathsf{w}}$ of the hard relation, and outputs a new response value $s \in \mathcal{D}_{\mathsf{resp}}$.
3. Finally, for witness extraction, we define $f_{\mathsf{ext}} \colon \mathcal{D}_{\mathsf{resp}} \times \mathcal{D}_{\mathsf{resp}} \to \mathcal{D}_{\mathsf{w}}$ that takes as input two response values $\tilde{s}, s \in \mathcal{D}_{\mathsf{resp}}$ and outputs a witness $y \in \mathcal{D}_{\mathsf{w}}$.

Our transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ is shown in Fig. 3.

```
$\mathsf{pSign}_{sk}(m, Y)$
1 : $(R_{\mathsf{pre}}, St) \leftarrow \mathsf{P}_1(sk)$
2 : $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$
3 : $h := \mathcal{H}(R_{\mathsf{sign}}, m)$
4 : $\tilde{s} \leftarrow \mathsf{P}_2(sk, R_{\mathsf{pre}}, h, St)$
5 : return $(h, \tilde{s})$
```

```
$\mathsf{pVrfy}_{pk}(m, Y; (h, \tilde{s}))$
1 : $\widehat{R}_{\mathsf{pre}} := \mathsf{V}_0(pk, h, \tilde{s})$
2 : $\widehat{R}_{\mathsf{sign}} := f_{\mathsf{shift}}(\widehat{R}_{\mathsf{pre}}, Y)$
3 : $b := (h = \mathcal{H}(\widehat{R}_{\mathsf{sign}}, m))$
4 : return $b$
```

```
$\mathsf{Adapt}_{pk}((h, \tilde{s}), y)$
1 : $s = f_{\mathsf{adapt}}(\tilde{s}, y)$
2 : return $(h, s)$

$\mathsf{Ext}_{pk}((h, s), (h, \tilde{s}), Y)$
1 : return $f_{\mathsf{ext}}(s, \tilde{s})$
```

Fig. 3: Generic transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to a $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ scheme

In order for $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ to be an adaptor signature scheme, we need the functions f_{shift}, f_{adapt} and f_{ext} to satisfy two properties. The first property is a homomorphic one and relates the functions f_{shift} and f_{adapt} to the commitment-recoverable component V_0 and the hard relation R. Informally, for all $(Y, y) \in \mathsf{R}$, we need the following to be equivalent: (i) Extract the public randomness from a response $\tilde{s}$ using V_0 and then apply f_{shift} to shift the public randomness by Y, and (ii) apply f_{adapt} to shift the *secret* randomness in $\tilde{s}$ by y and then extract the public randomness using V_0. Formally, for any public key pk, any challenge $h \in \mathsf{ChSet}$, any response value $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$ and any statement/witness pair $(Y, y) \in \mathsf{R}$, it must hold that:

$$f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)). \tag{1}$$

The second property requires that the function $f_{\mathsf{ext}}(\tilde{s}, \cdot)$ is the inverse function of $f_{\mathsf{adapt}}(\tilde{s}, \cdot)$ for any $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$. Formally, for any $y \in \mathcal{D}_{\mathsf{w}}$ and $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$, we have

$$y = f_{\mathsf{ext}}(f_{\mathsf{adapt}}(\tilde{s}, y), \tilde{s}). \tag{2}$$

To give an intuition about the functions f_{shift}, f_{adapt} and f_{ext} and their purpose, let us discuss their concrete instantiations for Schnorr signatures and show that they satisfy Equations (1) and (2). The instantiations for Katz-Wang signatures and Guillou-Quisquater signatures can be found in the full version of this paper [14].

Example 1 (Schnorr signatures). Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p where the discrete logarithm problem in $\mathbb{G}$ is hard. The functions IGen, P_1, P_2 and V_0 for Schnorr's signature scheme are defined in Fig. 4.

$\mathsf{IGen}(n)$	$\mathsf{P}_1(sk)$	$\mathsf{P}_2(sk, R, h, r)$	$\mathsf{V}_0(pk, h, s)$
$1: sk \leftarrow_{\$} \mathbb{Z}_q, pk = g^{sk}$	$1: r \leftarrow_{\$} \mathbb{Z}_q, R = g^r$	$1: s = r + h \cdot sk$	$1: R = g^s \cdot pk^{-h}$
$2:$ **return** (sk, pk)	$2:$ **return** (R, r)	$2:$ **return** s	$2:$ **return** (R)

Fig. 4: Schnorr signature scheme

Let us consider the hard relation $\mathsf{R} = \{(Y, y) \mid Y = g^y\}$, i.e., group elements and their discrete logarithms, and let us define the functions f_{shift}, f_{adapt}, f_{ext} as:

$$f_{\mathsf{shift}}(Y, R) := Y \cdot R, \quad f_{\mathsf{adapt}}(\tilde{s}, y) := \tilde{s} + y, \quad f_{\mathsf{ext}}(s, \tilde{s}) := s - \tilde{s}.$$

Intuitively, the function f_{shift} is *shifting* randomness in the group while the function f_{adapt} *shifts* randomness in the exponent. To prove that Eq. (1) holds, let us fix an arbitrary public key $pk \in \mathbb{G}$, a challenge $h \in \mathbb{Z}_q$, a response value $s \in \mathbb{Z}_q$ and a statement witness pair $(Y, y) \in \mathsf{R}$, i.e, $Y = g^y$. We have

$$\begin{aligned} f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, s), Y) &= f_{\mathsf{shift}}(g^s \cdot pk^{-h}, Y) = g^s \cdot pk^{-h} \cdot Y \\ &= g^{s+y} \cdot pk^{-h} = \mathsf{V}_0(pk, h, s + y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(s, y)) \end{aligned}$$

which is what we wanted to prove. In order to show that Eq. (2) holds, let us fix an arbitrary witness $y \in \mathbb{Z}_q$ and a response value $s \in \mathbb{Z}_q$. Then we have

$$f_{\mathsf{ext}}(f_{\mathsf{adapt}}(s, y), s) = f_{\mathsf{ext}}(s + y, s) = s + y - s = y$$

and hence Eq. (2) is satisfied as well.

We now show that the transformation from Fig. 3 is a secure adaptor signature scheme if functions $f_{\mathsf{shift}}, f_{\mathsf{adapt}}, f_{\mathsf{ext}}$ satisfying Equations (1) and (2) exist.

Theorem 2. *Assume that* $\mathsf{SIG}^{\mathsf{ID}}$ *is a* $\mathsf{SUF\text{-}CMA}$*-secure signature scheme transformed using Fig. 2, let* $f_{\mathsf{shift}}, f_{\mathsf{adapt}}$ *and* f_{ext} *be functions satisfying the relations from Equations* (1) *and* (2)*, and* R *be a hard relation. Then the resulting* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *scheme from the transformation in Fig. 3 is a secure adaptor signature scheme in the random oracle model.*

In order to prove Thm. 2, we must show that $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ satisfies *pre-signature correctness*, $\mathsf{aEUF\text{-}CMA}$ *security*, *pre-signature adaptability* and *witness extractability* properties described in Defs. 2 to 5 respectively.

Lemma 1 (Pre-Signature Correctness). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies pre-signature correctness as for Def. 2.*

Proof. Let us fix an arbitrary message m and a statement witness pair $(Y, y) \in \mathsf{R}$. Let $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$, $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$, $\sigma := \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$ and $y' := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$. From Fig. 3 we know that $\widetilde{\sigma} = (h, \tilde{s})$, $\sigma = (h, s)$ and $y' = f_{\mathsf{ext}}(s, \tilde{s})$, where we have $s := f_{\mathsf{adapt}}(\tilde{s}, y)$, $\tilde{s} \leftarrow \mathsf{P}_2(sk, R_{\mathsf{pre}}, h, St)$, $h := \mathcal{H}(R_{\mathsf{sign}}, m)$, $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$ and $(R_{\mathsf{pre}}, St) \leftarrow \mathsf{P}_1(sk)$. We first show $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$. From completeness of the ID scheme, we know that $\mathsf{V}_0(pk, h, \tilde{s}) = R_{\mathsf{pre}}$. Hence:

$$\mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m) = \mathcal{H}(f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), m) = \mathcal{H}(R_{\mathsf{sign}}, m) = h \tag{3}$$

which is what we needed to prove. We now show that $\mathsf{Vrfy}_{pk}(m; \sigma) = 1$. By Fig. 2, we need to show that $h = \mathcal{H}(\mathsf{V}_0(pk, h, s), m)$. This follows from the property of f_{shift}, f_{adapt} (cf. Eq. (1)) and Eq. (3) as follows:

$$\begin{aligned} \mathcal{H}(\mathsf{V}_0(pk, h, s), m) =& \mathcal{H}(\mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)), m) \\ &\overset{(1)}{=} \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m) \overset{(3)}{=} h. \end{aligned}$$

Finally, we need to show that $(Y, y') \in \mathsf{R}$. This follows from Eq. (2) since:

$$y' = f_{\mathsf{ext}}(s, \tilde{s}) = f_{\mathsf{ext}}(f_{\mathsf{adapt}}(\tilde{s}, y), \tilde{s}) \overset{(2)}{=} y.$$

Lemma 2 **(aEUF–CMA-Security).** *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the* aEUF–CMA *security as for Def. 3.*

Let us give first a high level overview of the proof. Our goal is to provide a reduction such that, given an adversary $\mathcal{A}$ who can win the experiment $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}$, we can build a simulator who can win the $\mathsf{strongSigForge}$ experiment of the underlying signature or can break the hardness of the relation R. In the first case, we check if $\mathcal{A}$'s forgery σ^* is equal to $\mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$. If so, we use $\mathcal{A}$ to break the hardness of the relation R by extracting the witness $y = \mathsf{Ext}(\sigma^*, \widetilde{\sigma}, Y)$. Otherwise, $\mathcal{A}$ was able to forge a signature "unrelated" to the pre-signature provided to it. In this case, it is used to win the $\mathsf{strongSigForge}$ experiment. All that remains is to answer $\mathcal{A}$'s signing and pre-signing queries using $\mathsf{strongSigForge}$'s signing queries. This is done by programming the random oracle such that the full-signatures generated by the challenger in the $\mathsf{strongSigForge}$ game look like pre-signatures for $\mathcal{A}$.

Proof. We prove the lemma by defining a series of game hops. The modifications for each game hop is presented in code form in the full version of this paper [14].

Game $\boldsymbol{G_0}$: This game is the original $\mathsf{aSigForge}$ experiment, where the adversary $\mathcal{A}$ outputs a valid forgery σ^* for a message m of its choice, while having access to pre-signing and signing oracles $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$ respectively. Being in the random oracle model, all the algorithms of the scheme and the adversary have access to the random oracle $\mathcal{H}$. Since $\boldsymbol{G_0}$ corresponds to $\mathsf{aSigForge}$, it follows that $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}(n) = 1] = \Pr[\boldsymbol{G_0} = 1]$.

Game $\boldsymbol{G_1}$: This game works as $\boldsymbol{G_0}$ except when the adversary outputs a forgery σ^*, the game checks if adapting the pre-signature $\widetilde{\sigma}$ using the secret witness y results in σ^*. If so, the game aborts.

Claim. Let Bad_1 be the event where $\boldsymbol{G_1}$ aborts. Then $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: This claim is proven by a reduction to the relation R. We construct a simulator $\mathcal{S}$ which breaks the hardness of R using $\mathcal{A}$ that causes $\boldsymbol{G_1}$ to abort with non-negligible probability. The simulator receives a challenge Y^*, and generates a key pair $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ in order to simulate $\mathcal{A}$'s queries to the oracles $\mathcal{H}$, $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$. This simulation of the oracles work as described in $\boldsymbol{G_1}$.

Upon receiving the challenge message m^* from $\mathcal{A}$, $\mathcal{S}$ computes a pre-signature $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$ and returns the pair $(\widetilde{\sigma}, Y)$ to the adversary. Upon $\mathcal{A}$ outputting a forgery σ^* and assuming that Bad_1 happened (i.e., $\mathsf{Adapt}(\widetilde{\sigma}, y) = \sigma$), pre-signature correctness (Def. 2) implies that the simulator can extract y^* by executing $\mathsf{Ext}(\sigma^*, \widetilde{\sigma}, Y^*)$ in order to obtain $(Y^*, y^*) \in \mathsf{R}$.

We note that the view of $\mathcal{A}$ in this simulation and in $\boldsymbol{G_1}$ are indistinguishable, since the challenge Y^* is an instance of the hard relation R and has the same distribution to the public output of GenR. Therefore, the probability that $\mathcal{S}$ breaks the hardness of R is equal to the probability that the event Bad_1 happens. Hence, we conclude that Bad_1 only happens with negligible probability. ■

Since games $\boldsymbol{G_1}$ and $\boldsymbol{G_0}$ are equivalent except if event Bad_1 occurs, it holds that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_1} = 1] + \nu_1(n)$.

Game $\boldsymbol{G_2}$: This game is similar to the previous game except for a modification in the $\mathcal{O}_{\mathrm{pS}}$ oracle. After the execution of $\mathsf{preSign}_{sk}$, the oracle obtains a pre-signature $\widetilde{\sigma}$ from which it extracts the randomness $R_{\mathsf{pre}} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$. The oracle computes $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$ and checks if $\mathcal{H}$ was already queried on the inputs $R_{\mathsf{pre}} \| m$ or $R_{\mathsf{sign}} \| m$ *before* the execution of pSign_{sk}. In this case the game aborts.

Claim. Let Bad_2 be the event that $\boldsymbol{G_2}$ aborts in $\mathcal{O}_{\mathrm{pS}}$. Then $\Pr[\mathsf{Bad}_2] \leq \nu_2(n)$, where ν_2 is a negligible function in n.

Proof: We first recall that the output of P_1 (i.e., R_{pre}) is uniformly random from a super-polynomial set of size q in the security parameter. From this it follows that R_{sign} is distributed uniformly at random in the same set. Furthermore, $\mathcal{A}$ being a PPT algorithm, it can only make polynomially many queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$ oracles. Denoting ℓ as the total number of queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$, we have: $\Pr[\mathsf{Bad}_2] =$

$\Pr[H'[R_{\mathsf{pre}}||m] \neq \bot \vee H'[R_{\mathsf{sign}}||m] \neq \bot] \leq 2\frac{\ell}{q} \leq \nu_2(n)$. This follows from the fact that ℓ is polynomial in the security parameter. ■

Since games $\boldsymbol{G_2}$ and $\boldsymbol{G_1}$ are identical except in the case where Bad_2 occurs, it holds that $\Pr[\boldsymbol{G_1} = 1] \leq \Pr[\boldsymbol{G_2} = 1] + \nu_2(n)$.

Game $\boldsymbol{G_3}$: In this game, upon a query to the $\mathcal{O}_{\mathrm{pS}}$, the game produces a full-signature instead of a pre-signature by executing Sign_{sk} instead of $\mathsf{preSign}_{sk}$. Accordingly, it programs the random oracle $\mathcal{H}$ to make the full-signature "look like" a pre-signature from the point of view of the adversary $\mathcal{A}$. This is done by:

1. It sets $\mathcal{H}(R_{\mathsf{pre}}\|m)$ to the value stored at position $\mathcal{H}(R_{\mathsf{sign}}\|m)$.
2. It sets $\mathcal{H}(R_{\mathsf{sign}}\|m)$ to a fresh value chosen uniformly at random.

The above programming makes sense as our definition of f_{shift} requires it to be deterministic and to possess the same domain and codomain with respect to the commitment set $\mathcal{D}_{\mathsf{rand}}$. Note further that $\mathcal{A}$ can only notice that $\mathcal{H}$ was programmed if it was previously queried on either $R_{\mathsf{pre}}\|m$ or $R_{\mathsf{sign}}\|m$. But as described in the previous game, we abort if such an event happens. Hence, we have that $\Pr[\boldsymbol{G_2} = 1] = \Pr[\boldsymbol{G_3} = 1]$.

Game $\boldsymbol{G_4}$: In this game, we impose new checks during the challenge phase that are same as the ones imposed in $\boldsymbol{G_2}$ during the execution of $\mathcal{O}_{\mathrm{pS}}$.

Claim. Let Bad_3 be the event that $\boldsymbol{G_4}$ aborts in the challenge phase. Then $\Pr[\mathsf{Bad}_3] \leq \nu_3(n)$, where ν_3 is a negligible function in n.

Proof: The proof is identical to the proof in $\boldsymbol{G_2}$. ■

It follows that $\Pr[\boldsymbol{G_4} = 1] \leq \Pr[\boldsymbol{G_3} = 1] + \nu_3(n)$.

Game $\boldsymbol{G_5}$: Similar to game $\boldsymbol{G_3}$, we generate a signature instead of a pre-signature in the challenge phase and program $\mathcal{H}$ such that the full-signature looks like a correct pre-signature from $\mathcal{A}$'s point of view. We get $\Pr[\boldsymbol{G_5} = 1] = \Pr[\boldsymbol{G_4} = 1]$.

Now that the transition from the original $\mathsf{aSigForge}$ experiment (game $\boldsymbol{G_0}$) to game $\boldsymbol{G_5}$ is indistinguishable, it only remains to show the existence of a simulator $\mathcal{S}$ that can perfectly simulate $\boldsymbol{G_5}$ and uses $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game. The modifications from games $\boldsymbol{G_1}$ - $\boldsymbol{G_5}$ and the simulation in code form can be found in the full version of this paper [14].

We emphasize that the main differences between the simulation and Game $\boldsymbol{G_5}$ are syntactical. Namely, instead of generating the public and secret keys and computing the algorithm Sign_{sk} and the random oracle $\mathcal{H}$, $\mathcal{S}$ uses its oracles $\mathsf{SIG}^{\mathsf{ID}}$ and $\mathcal{H}^{\mathsf{ID}}$. Therefore, $\mathcal{S}$ perfectly simulates $\boldsymbol{G_5}$. It remains to show that $\mathcal{S}$ can use the forgery output by $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game.

Claim. (m^*, σ^*) constitutes a valid forgery in game $\mathsf{strongSigForge}$.

Proof: To prove this claim, we show that the tuple (m^*, σ^*) has not been returned by the oracle $\mathsf{SIG}^{\mathsf{ID}}$ before. First note that $\mathcal{A}$ wins the experiment if it has not queried on the challenge message m^* to $\mathcal{O}_{\mathrm{pS}}$ or $\mathcal{O}_{\mathrm{S}}$. Therefore, $\mathsf{SIG}^{\mathsf{ID}}$ is queried on m^* only during the challenge phase. If $\mathcal{A}$ outputs a forgery σ^* that is equal to the signature σ as output by $\mathsf{SIG}^{\mathsf{ID}}$, it would lose the game since this signature is not valid given the fact that $\mathcal{H}$ is programmed.

Hence, $\mathsf{SIG}^{\mathsf{ID}}$ has never output σ^* when queried on m^* before, thus making (m^*, σ^*) a valid forgery for game $\mathsf{strongSigForge}$. ■

From games $\boldsymbol{G_0} - \boldsymbol{G_5}$, we have that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_5} = 1] + \nu(n)$, where $\nu(n) = \nu_1(n) + \nu_2(n) + \nu_3(n)$ is a negligible function in n. Since $\mathcal{S}$ simulates game $\boldsymbol{G_5}$ perfectly, we also have that $\Pr[\boldsymbol{G_5} = 1] = \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}}(n) = 1]$. Combining this with the probability statement in $\boldsymbol{G_0}$, we obtain the following:

$\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}(n) = 1] \leq \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}^{\mathsf{ID}}}(n) = 1] + \nu(n)$.

Recall that the negligible function $\nu_1(n)$, contained in the sum $\nu(n)$ above, precisely quantifies the adversary's advantage in breaking the hard relation R. Thus, the probability of breaking the unforgeability of the $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ is clearly bounded above by that of breaking either R or the strong unforgeability of $\mathsf{SIG}^{\mathsf{ID}}$.

Lemma 3 (Pre-Signature Adaptability). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the pre-signature adaptability as for Def. 4.*

Proof. Assume $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$, with the notations having their usual meanings from Fig. 3, which means $h = \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m)$. For any valid pair $(Y, y) \in R$, we can use the homomorphic property from Eq. (1). Then, for such a pair $(Y, y) \in R$, plugging $f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y))$ in the above equation implies $h = \mathcal{H}(\mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)), m)$. This directly implies $\mathsf{Vrfy}_{pk}(m; \sigma) = 1$, where $s = f_{\mathsf{adapt}}(\tilde{s}, y)$ and $\sigma = (h, s)$. Therefore, adapting the valid pre-signature would also result in a valid full-signature.

Lemma 4 (Witness Extractability). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the witness extractability as for Def. 5.*

This proof is very similar to the proof of Lemma 2 with the mere difference that we only need to provide a reduction to the strongSigForge experiment. This is because in the $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R}_g,\mathsf{SIG}^{\mathsf{ID}}}}$ experiment, $\mathcal{A}$ provides the public value Y^* and must forge a valid full-signature σ^* such that $(Y^*, \mathsf{Ext}_{pk}(\sigma^*, \widetilde{\sigma}, Y^*)) \notin \mathsf{R}$. The full proof can be found in the full version of this paper [14].

Remark 2. We note that our proofs for the aEUF–CMA security and witness extractability are in its essence reductions to the strong unforgeability of the underlying signature schemes. Yet the Fiat-Shamir transformation does not immediately guarantee the resulting signature scheme to be strongly unforgeable. However, we first note that many such signature schemes are indeed strongly unforgeable, for instance Schnorr [25], Katz-Wang (from Chaum-Pedersen identification scheme) [24] and Guillou-Quisquater [1] signature schemes all satisfy strong unforgeability. Moreover, one can transform any Fiat-Shamir based existentially unforgeable signature scheme into a strongly unforgeable one via the generic transformation using the results of Bellare et.al. [4].

4 Two-party Signatures with Aggregatable Public Keys from Identification Schemes

Before providing our definition and generic transformation for two-party adaptor signatures, we show how to generically transform signature schemes based on identification schemes into two-party signature schemes with aggregatable public keys denoted by SIG_2. In Sec. 5, we then combine the techniques used in this section with the ones from Sec. 3 in order to generically transform identification schemes into two-party adaptor signature schemes.

Informally, a SIG_2 scheme allows two parties to jointly generate a signature which can be verified under their combined public keys. An application of such signature schemes can be found in cryptocurrencies where two parties wish to only allow conditional payments such that both users have to sign a transaction in order to spend some funds. Using SIG_2, instead of submitting two separate signatures, the parties can submit a single signature while enforcing the same condition (i.e., a transaction must have a valid signature under the combined key) and hence reduce the communication necessary with the blockchain. Importantly and unlike threshold signature schemes, the key generation here is non-interactive. In other words, parties generate their public and secret keys independently and anyone who knows both public keys can compute the joint public key of the two parties.

We use the notation $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}$ to represent a two-party interactive protocol Func between P_i and P_{1-i} with respective secret inputs x_i, x_{1-i} for $i \in \{0, 1\}$. Furthermore, if there are common public inputs e.g., $y_1, \cdots, y_n$ we use the notation $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}(y_1, \cdots, y_n)$. We note that the execution of a protocol might not be symmetric, i.e., party $\mathcal{P}_i$ executes the procedures $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}$ while party $\mathcal{P}_{1-i}$ executes the procedures $\Pi_{\mathsf{Func}\langle x_{1-i}, x_i\rangle}$.

4.1 Two-party Signatures with Aggregatable Public Keys

We start with defining a two-party signature scheme with aggregatable public keys. Our definition is inspired by the definitions from prior works [8, 26, 7].

Definition 7 (Two-party Signature with Aggregatable Public Keys). *A two-party signature scheme with aggregatable public keys is a tuple of PPT protocols and algorithms* $\mathsf{SIG}_2 = (\mathsf{Setup}, \mathsf{Gen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$*, formally defined as:*

$\mathsf{Setup}(1^n)$**:** *is a PPT algorithm that on input a security parameter n, outputs public parameters pp.*
$\mathsf{Gen}(pp)$**:** *is a PPT algorithm that on input public parameter pp, outputs a key pair (sk, pk).*
$\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_0, pk_1, m)$**:** *is an interactive, PPT protocol that on input secret keys sk_i from party $\mathcal{P}_i$ with $i \in \{0,1\}$ and common values $m \in \{0,1\}^*$ and pk_0, pk_1, outputs a signature σ.*
$\mathsf{KAg}(pk_0, pk_1)$**:** *is a DPT algorithm that on input two public keys pk_0, pk_1, outputs an aggregated public key apk.*
$\mathsf{Vrfy}_{apk}(m;\sigma)$**:** *is a DPT algorithm that on input public parameters pp, a public key apk, a message $m \in \{0,1\}^*$ and a signature σ, outputs a bit b.*

The *completeness* property of SIG_2 guarantees that if the protocol Π_{Sign} is executed correctly between the two parties, the resulting signature is a valid signature under the aggregated public key.

Definition 8 (Completeness). *A two-party signature with aggregatable public keys SIG_2 satisfies completeness, if for all key pairs $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ and messages $m \in \{0,1\}^*$, the protocol $\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_0, pk_1, m)$ outputs a signature σ to both parties $\mathcal{P}_0, \mathcal{P}_1$ such that $\mathsf{Vrfy}_{apk}(m;\sigma) = 1$ where $apk := \mathsf{KAg}(pk_0, pk_1)$.*

A two-party signature scheme with aggregatable public keys should satisfy *unforgeability.* At a high level, this property guarantees that if one of the two parties is malicious, this party is not able to produce a valid signature under the aggregated public key without cooperation of the other party. We formalize the property through an experiment $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}$, where $b \in \{0,1\}$ defines which of the two parties is corrupt. This experiment is initialized by a security parameter n and run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which proceeds as follows. The challenger first generates the public parameters pp by running the setup procedure $\mathsf{Setup}(1^n)$ as well as a signing key pair (sk_{1-b}, pk_{1-b}) by executing $\mathsf{Gen}(1^n)$, thereby simulating the honest party $\mathcal{P}_{1-b}$. Thereafter, $\mathcal{C}$ forwards pp_{C} and pk_{1-b} to the adversary $\mathcal{A}$ who generates its own key pair (sk_b, pk_b), thereby emulating the malicious party P_b, and submits (sk_b, pk_b) to $\mathcal{C}$. The adversary $\mathcal{A}$ additionally obtains access to an *interactive* and stateful signing oracle $\mathcal{O}^b_{\Pi_{\mathrm{S}}}$, which simulates the honest party $\mathcal{P}_{1-b}$ during the execution of $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b},\cdot\rangle}$. Furthermore, every queried message m is stored in a query list $\mathcal{Q}$.

Eventually, $\mathcal{A}$ outputs a forgery in form of a $\mathsf{SIG}_2^{\mathsf{ID}}$ signature σ^* and a message m^*. $\mathcal{A}$ wins the experiment if σ^* is a valid signature for m^* under the aggregated public key $apk := \mathsf{KAg}(pk_0, pk_1)$ and m^* was never queried before, i.e., $m^* \notin \mathcal{Q}$. Below, we give a formal definition of the unforgeability game.

Definition 9 (2-EUF–CMA Security). *A two-party, public key aggregatable signature scheme SIG_2 is unforgeable if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that: for $b \in \{0,1\}$, $\Pr[\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n) = 1] \leq \nu(n)$, where the experiment $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n)$ is defined as follows:*

```
$\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $(\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_{\mathrm{S}}}(\cdot)}(pk_{1-b}, sk_b, pk_b)$
5 : return $\left(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{\mathsf{KAg}(pk_0, pk_1)}(m^*;\sigma^*)\right)$

$\mathcal{O}^b_{\Pi_{\mathrm{S}}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b},\cdot\rangle}(pk_0, pk_1, m)$
```

Remark 3 (On security definition.). There are two different approaches for modeling signatures with aggregatable public keys in the literature, namely the plain public-key model [3] (also known as key-verification model [12]) and the knowledge-of-secret-key (KOSK) model [7]. In the plain public-key setting the adversary chooses a key pair (sk_b, pk_b) and only declares the public key pk_b to the challenger in the security game. However, security proofs in this setting typically require rewinding techniques with the forking lemma. This is undesirable for the purpose of this paper, as we aim to construct adaptor signatures and its two-party variant generically as building blocks for further applications such as payment channels [2]. Payment channels are proven secure in the UC framework that does not allow the use of rewinding techniques in order to ensure concurrency. Thus, the plain public-key model does not seem suitable for our purpose. In the KOSK setting, however, the adversary outputs its (possibly maliciously chosen) key pair (sk_b, pk_b) to the challenger. In practice this means that the parties need to exchange zero-knowledge proofs of knowledge of their secret key[3]. Similar to previous works [7, 27], we do not require the forking lemma or rewinding in the KOSK setting and hence follow this approach.

4.2 Generic Transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{SIG}^{\mathsf{ID}}_2$

We now give a generic transformation from $\mathsf{SIG}^{\mathsf{ID}}$ schemes to two-party signature schemes with aggregatable public keys.

At a high level, our transformation turns the signing procedure into an interactive protocol which is executed between the two parties $\mathcal{P}_0, \mathcal{P}_1$. The main idea is to let both parties engage in a randomness exchange protocol in order to generate a joint public randomness which can then be used for the signing procedure. In a bit more detail, to create a joint signature, each party $\mathcal{P}_i$ for $i \in \{0, 1\}$ can individually create a partial signature with respect to the *joint randomness* by using the secret key sk_i and exchange her partial signature with $\mathcal{P}_{1-i}$. The joint randomness ensures that both partial signatures can be combined to one jointly computed signature.

In the following, we describe the randomness exchange protocol that is executed during the signing procedure in more detail, as our transformation heavily relies on it. The protocol, denoted by $\Pi_{\mathsf{Rand\text{-}Exc}}$, makes use of two cryptographic building blocks, namely an extractable commitment scheme $\mathsf{C} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Extract})$ and a NIZK proof system $\mathsf{NIZK} = (\mathsf{Setup}_{\mathsf{R}}, \mathsf{Prove}, \mathsf{Verify})$. Consequently, the common input to both parties $\mathcal{P}_0$ and $\mathcal{P}_1$ are the public parameters pp_{C} of the commitment scheme, while each party P_i takes as secret input her secret key sk_i. In the following, we give description of the $\Pi_{\mathsf{Rand\text{-}Exc}\langle sk_0, sk_1\rangle}(pp_{\mathsf{C}}, \mathsf{crs})$ protocol and present it in a concise way in Fig. 5.

$\mathcal{P}_0(pp_{\mathsf{C}}, \mathsf{crs}, sk_0)$		$\mathcal{P}_1(pp_{\mathsf{C}}, \mathsf{crs}, sk_1)$
$(R_0, St_0) \leftarrow \mathsf{P}_1(sk_0)$		
$\pi_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_0, sk_0)$		
$(c, d) \leftarrow \mathsf{C.Com}(pp_{\mathsf{C}}, (R_0, \pi_0))$	$\xrightarrow{c}$	$(R_1, St_1) \leftarrow \mathsf{P}_1(sk_1)$
	$\xleftarrow{R_1, \pi_1}$	$\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_1, sk_1)$
	$\xrightarrow{d}$	$R_0' \leftarrow \mathsf{C.Dec}(pp_{\mathsf{C}}, c, d)$
If $\mathsf{NIZK.Verify}(\mathsf{crs}, R_1, \pi_1) = 0$, then abort		If $\mathsf{NIZK.Verify}(\mathsf{crs}, R_0', \pi_0) = 0$, then abort
R_0, St_0, R_1		R_1, St_1, R_0

Fig. 5: $\Pi_{\mathsf{Rand\text{-}Exc}}$ Protocol

1. Party $\mathcal{P}_0$ generates her public randomness R_0 using algorithm P_1 from the underlying ID scheme alongside a NIZK proof $\pi_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_0, sk_0)$ that this computation was executed correctly with the corresponding secret value sk_0. $\mathcal{P}_0$ executes $(c, d) \leftarrow \mathsf{C.Com}(pp, (R_0, \pi_0))$ to commit to R_0 and π_0 and sends the commitment c to $\mathcal{P}_1$.

[3] Using techniques from [20, 16] it is possible to obtain NIZKs which allow for witness extraction without rewinding.

```
$\mathsf{Setup}(1^n)$
  1 : $pp_{\mathsf{C}} \leftarrow \mathsf{C.Gen}(1^n)$
  2 : $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}_{\mathsf{R}}(1^n)$
  3 : $\mathbf{return}\ pp := (1^n, pp_{\mathsf{C}}, \mathsf{crs})$

$\mathsf{Gen}(pp)$
  1 : Parse $pp = (1^n, pp_{\mathsf{C}}, \mathsf{crs})$
  2 : $(sk, pk') \leftarrow \mathsf{IGen}(n)$
  3 : $pk := (pp, pk')$
  4 : $\mathbf{return}\ (sk, pk)$

$\mathsf{KAg}(pk_0, pk_1)$
  1 : $apk := f_{\mathsf{com\text{-}pk}}(pk_0, pk_1)$
  2 : $\mathbf{return}\ apk$

$\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_i, pk_{1-i}, m)$
  1 : Parse $pk_i = ((1^n, pp_{\mathsf{C}}, \mathsf{crs}), pk'_i)$
  2 : $(R_i, St_i, R_{1-i}) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_i, sk_{1-i}\rangle}(pp_{\mathsf{C}}, \mathsf{crs})$
  3 : $R_{\mathsf{sign}} := f_{\mathsf{com\text{-}rand}}(R_0, R_1)$
  4 : $h := \mathcal{H}(R_{\mathsf{sign}}, m)$
  5 : $s_i \leftarrow \mathsf{P}_2(sk_i, R_i, h, St_i)$
  6 : $s_{1-i} \leftarrow \Pi_{\mathsf{Exchange}}\langle s_i, s_{1-i}\rangle$
  7 : $(h, s) := f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1))$
  8 : $\mathbf{return}\ (h, s)$

$\mathsf{Vrfy}_{apk}(m; (h, s))$
  1 : $R_{\mathsf{sign}} := \mathsf{V}_0(apk, h, s)$
  2 : $\mathbf{return}\ h := \mathcal{H}(R_{\mathsf{sign}}, m)$
```

Fig. 6: $\mathsf{SIG}_2^{\mathsf{ID}}$: SIG_2 scheme from identification scheme.

2. Upon receiving the commitment c from $\mathcal{P}_0$, party $\mathcal{P}_1$ generates her public randomness R_1 using algorithm P_1. She also computes a NIZK proof as $\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_1, sk_1)$, which proves correct computation of R_1, and sends R_1 and π_1 to $\mathcal{P}_0$.
3. Upon receiving R_1 and π_1 from $\mathcal{P}_1$, $\mathcal{P}_0$ sends the opening d to her commitment c to $\mathcal{P}_1$.
4. $\mathcal{P}_1$ opens the commitment in this round. At this stage, both parties check that the received zero-knowledge proofs are valid. If the proofs are valid, each party $\mathcal{P}_i$ for $i \in \{0, 1\}$ outputs R_i, St_i, R_{1-i}.

Our transformation can be found in Fig. 6. Note that we use a deterministic function $f_{\mathsf{com\text{-}rand}}(\cdot, \cdot)$ in step 3 in the signing protocol which combines the two public random values R_0 and R_1. In step 6 of the same protocol, we assume that the partial signatures are exchanged between the parties via the protocol Π_{Exchange} upon which the parties can combine them using a deterministic function $f_{\mathsf{com\text{-}sig}}(\cdot, \cdot)$ in step 7. Further, a combined signature can be verified under a combined public key of the two parties. In more detail, to verify a combined signature $(h, s) := f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1))$, in step 7, there must exist an additional deterministic function $f_{\mathsf{com\text{-}pk}}(\cdot, \cdot)$ (in step 1 of the KAg algorithm) such that:

$$\Pr\left[\mathsf{Vrfy}_{apk}(m; (h, s)) = 1 \;\middle|\; \begin{array}{l} (pk_0, sk_0) \leftarrow \mathsf{IGen}(n), (pk_1, sk_1) \leftarrow \mathsf{IGen}(n) \\ (h, s) \leftarrow \Pi_{\mathsf{Sign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m) \\ apk := f_{\mathsf{com\text{-}pk}}(pk_0, pk_1) \end{array}\right] = 1. \tag{4}$$

We also require that given a full signature and a secret key sk_i with $i \in \{0, 1\}$, it is possible to extract a valid partial signature under the the public key pk_{1-i} of the other party. In particular, there exists a function $f_{\mathsf{dec\text{-}sig}}(\cdot, \cdot, \cdot)$ such that:

$$\Pr\left[\mathsf{Vrfy}_{pk_{1-i}}(m; (h, s_{1-i})) = 1 \;\middle|\; \begin{array}{l} (pk_0, sk_0) \leftarrow \mathsf{IGen}(n), (pk_1, sk_1) \leftarrow \mathsf{IGen}(n) \\ (h, s) \leftarrow \Pi_{\mathsf{Sign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m) \\ (h, s_{1-i}) := f_{\mathsf{dec\text{-}sig}}(sk_i, pk_i, (h, s)) \end{array}\right] = 1. \tag{5}$$

Note that equations 4 and 5 implicitly define $f_{\mathsf{com\text{-}sig}}$ through the execution of Π_{Sign} in the conditional probabilities.

The instantiations of these functions for Schnorr, Katz-Wang signatures and Guillou-Quisquater signatures can be found in the full version of this paper [14].

We note the similarity between this transformation with that in Fig. 3. In particular, both of them compute the public randomness R_{sign} by shifting the original random values. Note also that running the algorithm V_0 on the inputs (pk_i, h, s_i) would return $R_i, \forall i \in \{0, 1\}$.

Below, we show that the transformation in Fig. 6 provides a secure two-party signature with aggregatable public keys. To this end, we show that $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies SIG_2 *completeness* and *unforgeability* from Def. 8 and Def. 9, respectively.

Theorem 3. *Assume that $\mathsf{SIG}^{\mathsf{ID}}$ is an $\mathsf{SUF\text{-}CMA}$-secure signature scheme transformed using Fig. 2. Further, assume that the functions $f_{\mathsf{com\text{-}sig}}$, $f_{\mathsf{com\text{-}pk}}$ and $f_{\mathsf{dec\text{-}sig}}$ satisfy the relations, Equations (4) and (5) respectively. Let NIZK be a non-interactive zero-knowledge proof system and let C be an extractable commitment scheme. Then the resulting $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme from the transformation in Fig. 6 is a secure two-party signature scheme with aggregatable public keys in the random oracle model.*

Lemma 5. *Under the assumptions of Thm. 3, $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies Def. 8.*

Proof. The proof follows directly from Eq. 4 and the construction of KAg algorithm in Fig. 6.

Lemma 6. *Under the assumptions of Thm. 3, $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies Def. 9.*

Proof. We prove this lemma via reduction to the unforgeability property of the $\mathsf{SIG}^{\mathsf{ID}}$ scheme. That is, we show that if there exists an adversary $\mathcal{A}$ that breaks the unforgeability property of the $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme with more than negligible probability, then we can construct an adversary $\mathcal{S}$ that breaks the unforgeability of the $\mathsf{SIG}^{\mathsf{ID}}$ scheme with more than negligible probability. We show a series of games, starting with $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}$, such that each game is computationally indistinguishable from the previous one. The last game is modified in such a way that $\mathcal{S}$ can use $\mathcal{A}$'s forgery to create its own forgery for the unforgeability game against the $\mathsf{SIG}^{\mathsf{ID}}$ scheme. We prove Lemma 6 by separately considering the cases of the adversary corrupting party $\mathcal{P}_0$ or party $\mathcal{P}_1$, respectively.

Adversary corrupts $\mathcal{P}_0$. In the following we give the proof in case the adversary corrupts party $\mathcal{P}_0$.
Game G_0: This is the regular $\mathsf{SigForge}^0_{\mathcal{A},\mathsf{SIG}_2}(n)$ experiment, in which the adversary plays the role of party $\mathcal{P}_0$. In the beginning, the challenger generates the public parameters as $pp \leftarrow \mathsf{Setup}(1^n)$. Note that the Setup procedure, apart from computing $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup_R}(1^n)$, includes the execution of $\mathsf{C.Gen}$ through which the challenger can learn the trapdoor tr for the commitment scheme C. Further, the challenger generates a fresh signing key pair $(sk_1, pk_1) \leftarrow \mathsf{Gen}(1^n)$, sends pp and pk_1 to $\mathcal{A}$ and receives the adversary's key pair (pk_0, sk_0).

Game G_1: This game proceeds exactly like the previous game, with a modification in the signing oracle. Upon $\mathcal{A}$ querying the signing oracle on message m, it sends the commitment c to its public randomness R_0. The challenger, using the trapdoor tr, then extracts a randomness $R_0' \leftarrow \mathsf{C.Extract}(pp, tr, c)$ and computes the joint randomness as $R_{\mathsf{sign}} \leftarrow f_{\mathsf{com\text{-}rand}}(R_0', R_1)$. Upon receiving the opening d to c from the adversary, the challenger checks if $R_0' = \mathsf{C.Dec}(pp, c, d)$. If this does not hold, the challenger aborts.

Claim. Let Bad_1 be the event that G_1 aborts in the signing oracle. Then, we have $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: Note that the challenger in game G_1 aborts only if the extracted value R_0' from commitment c is not equal to the actual committed value R_0 in c, i.e., if $\mathsf{C.Extract}(pp, tr, c) \neq \mathsf{C.Dec}(pp, c, d)$. By the extractability property of C this happens only with negligible probability. In other words, it holds that $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n. ■

Game G_2: This game proceeds as game G_1, with a modification to the signing oracle. Upon $\mathcal{A}$ querying the signing oracle on input message m, instead of generating its signature (h, s_1) with respect to the joint public randomness R_{sign}, the challenger generates it only with respect to its own randomness R_1. That is, the challenger computes $h := \mathcal{H}(R_1, m)$ and executes the signing procedure w.r.t. this h. The challenger then checks if the adversary has queried the random oracle on input (R_{sign}, m) or (R_1, m) before the signing query. If so, the challenger aborts. Otherwise, the challenger programs the random oracle such that on input (R_{sign}, m) it returns h.

Claim. Let Bad_2 be the event that $\boldsymbol{G_2}$ aborts in the signing oracle. Then, we have $\Pr[\mathsf{Bad}_2] \leq \nu_2(n)$, where ν_2 is a negligible function in n.

Proof: Note that R_1 and R_{sign} are uniformly random elements from a set of size q. Furthermore, $\mathcal{A}$ being a PPT algorithm, can only make polynomially many queries to $\mathcal{H}$ and $\mathcal{O}_{\mathrm{pS}}$ oracles. Denoting by ℓ the total number of queries to $\mathcal{H}$ and $\mathcal{O}_{\mathrm{S}}$, we have: $\Pr[\mathsf{Bad}_2] = \Pr[\mathcal{H}(R_{\mathsf{sign}}, m) \neq \bot] \leq \frac{\ell}{q} \leq \nu_2(n)$. ■

Game $\boldsymbol{G_3}$: In this game, the only modification as compared to the previous game is that during the Setup procedure, the challenger executes the algorithm $(\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{NIZK.Setup}'_{\mathsf{R}}(1^n)$ instead of $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{R}}(1^n)$, which allows the challenger to learn the trapdoor τ. Since the two distributions $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{R}}(1^n)\}$ and $\{\widetilde{\mathsf{crs}} : (\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{Setup}'_{\mathsf{R}}(1^n)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability, we have that $\Pr[\boldsymbol{G_2} = 1] \leq \Pr[\boldsymbol{G_3} = 1] + \nu_3(n)$ where ν_3 is a negligible function in n.

Game $\boldsymbol{G_4}$: This game proceeds exactly like the previous game with a modification in the signing oracle. Instead of computing the NIZK proof π_1, the challenger executes $\pi_{\mathsf{S}} \leftarrow \mathsf{S}(\widetilde{\mathsf{crs}}, \tau, R_1)$. Due to the zero-knowledge property of the NIZK proof system, we know that the distributions $\{\pi : \pi \leftarrow \mathsf{Prove}(\widetilde{\mathsf{crs}}, R_1, r_1)\}$ and $\{\pi_{\mathsf{S}} : \pi_{\mathsf{S}} \leftarrow \mathsf{S}(\widetilde{\mathsf{crs}}, \tau, R_1)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability. Therefore, it holds that $\Pr[\boldsymbol{G_3} = 1] \leq \Pr[\boldsymbol{G_4} = 1] + \nu_4(n)$ where ν_4 is a negligible function in n.

Having shown that the transition from game $\boldsymbol{G}_0$ to game $\boldsymbol{G}_4$ is indistinguishable, it remains to show that we can construct an adversary $\mathcal{S}$ playing in game $\mathsf{strongSigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}$, which simulates game $\boldsymbol{G}_4$ to $\mathcal{A}$ and which can use $\mathcal{A}$'s forgery to win its own game. The simulation of $\mathcal{S}$ differs from game $\boldsymbol{G}_4$ as follows:

1. $\mathcal{S}$ does not choose its own key pair, but instead uses the public key that it receives from game $\mathsf{strongSigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}$.
2. Upon a signing oracle query on message m, $\mathcal{S}$ does not compute its signature s_1, but instead queries its own signing oracle on message m.
3. Upon a random oracle query on message m, $\mathcal{S}$ forwards the query to its own random oracle

It is easy to see, that $\mathcal{S}$ perfectly simulates game $\boldsymbol{G}_4$. It remains to show that $\mathcal{S}$ can use a valid forgery output by $\mathcal{A}$ to win the $\mathsf{strongSigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}$ game.

Claim. A valid forgery $(m^*, (h^*, s^*))$ output by $\mathcal{A}$ in game $\mathsf{SigForge}^1_{\mathcal{A},\mathsf{SIG}_2^{\mathsf{ID}}}$ can be transformed into a valid forgery $(m^*, (h^*, s_1^*))$ in game $\mathsf{strongSigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}$.

Proof: When $\mathcal{A}$ outputs $(m^*, (h^*, s^*))$, $\mathcal{S}$ extracts the partial signature (h^*, s_1^*) by executing $f_{\mathsf{dec\text{-}sig}}(sk_0, pk_0, (h^*, s^*))$ (from Eq. 5) and uses (h^*, s_1^*) as its own forgery. Note that $\mathcal{S}$ knows the adversary's key pair (sk_0, pk_0). By definition, $\mathcal{A}$ wins this game if it has not queried a signature on m^* before. Thus, $\mathcal{S}$ has also not queried the its own signing oracle on m^* before and $\mathcal{S}$ has not programmed the random oracle on any input $(\cdot, m^*)$. Finally, Eq. (5) implies that $(m^*, (h^*, s_1^*))$ is a valid forgery under the public key pk_1. ■

From games $\boldsymbol{G_0} - \boldsymbol{G_4}$, we have that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_4} = 1] + \nu(n)$, where $\nu(n) = \nu_1(n) + \nu_2(n) + \nu_3(n) + \nu_4(n)$ is a negligible function in n. Thus, we have $\Pr[\mathsf{SigForge}^1_{\mathcal{A},\mathsf{SIG}_2^{\mathsf{ID}}}(n) = 1] \leq \Pr[\mathsf{strongSigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}(n) = 1] + \nu(n)$.

Adversary corrupts $\mathcal{P}_1$. The proof for the case where the adversary corrupts $\mathcal{P}_1$, follows exactly the same steps as above with only a change in game $\boldsymbol{G_1}$. Instead of extracting the randomness R_0' from the commitment c, we must show that the adversary does not learn any information (except with negligible probability) about R_0 from c. We can do so by exhibiting a reduction to the hiding property of the commitment scheme C.

5 Two-party Aggregatable Adaptor Signatures

We are now ready to formally introduce the notion of two-party adaptor signatures with aggregatable public keys which we denote by aSIG_2. Our definition can be seen as a combination of the definition of adaptor signatures from Sec. 3 and the definition of two-party signatures with aggregatable public keys from Sec. 4. Unlike the single party adaptor signatures, in aSIG_2 both parties have the role of the signer and generate pre-signatures cooperatively. Furthermore, both parties can adapt the pre-signature given a witness value y. We note that both the pre-signature and the full-signature are valid under the aggregated public keys of the two parties. We formally define an aSIG_2 scheme w.r.t. a SIG_2 scheme (which is in turn defined w.r.t. a SIG scheme) and a hard relation R.

Afterwards, we show how to instantiate our new definition. Concretely, we present a generic transformation that turns a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme with certain homomorphic properties into a two-party adaptor signatures scheme. As a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme is constructed w.r.t. a $\mathsf{SIG}^{\mathsf{ID}}$ scheme (cf. Sec. 4), the construction presented in this section can implicitly transform digital signatures based on ID schemes to two-party adaptor signatures.

The definition of a two-party adaptor signature scheme aSIG_2 is similar to the definition of a standard adaptor signature scheme as for Def. 1. The main difference lies in the pre-signature generation. Namely, the algorithm pSign is replaced by a *protocol* Π_{pSign} which is executed between two parties.

Definition 10 (Two-Party Adaptor Signature Scheme with Aggregatable Public Keys). *A two-party adaptor signature scheme with aggregatable public keys is defined w.r.t. a hard relation* R *and a two-party signature scheme with aggregatable public keys* $\mathsf{SIG}_2 = (\mathsf{Setup}, \mathsf{Gen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$*. It is run between parties* $\mathcal{P}_0, \mathcal{P}_1$ *and consists of a tuple* $\mathsf{aSIG}_2 = (\Pi_{\mathsf{pSign}}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ *of efficient protocols and algorithms which are defined as follows:*

$\Pi_{\mathsf{pSign}\langle sk_i, sk_{1-i}\rangle}(pk_0, pk_1, m, Y)$**:** *is an interactive protocol that on input secret keys* sk_i *from party* $\mathcal{P}_i$ *with* $i \in \{0,1\}$ *and common values public keys* pk_i*, message* $m \in \{0,1\}^*$ *and statement* $Y \in L_{\mathsf{R}}$*, outputs a pre-signature* $\widetilde{\sigma}$*.*

$\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma})$**:** *is a DPT algorithm that on input an aggregated public key apk, a message* $m \in \{0,1\}^*$*, a statement* $Y \in L_{\mathsf{R}}$ *and a pre-signature* $\widetilde{\sigma}$*, outputs a bit b.*

$\mathsf{Adapt}_{apk}(\widetilde{\sigma}, y)$**:** *is a DPT algorithm that on input an aggregated public key apk, a pre-signature* $\widetilde{\sigma}$ *and witness* y*, outputs a signature* σ*.*

$\mathsf{Ext}_{apk}(\sigma, \widetilde{\sigma}, Y)$**:** *is a DPT algorithm that on input an aggregated public key apk, a signature* σ*, pre-signature* $\widetilde{\sigma}$ *and statement* $Y \in L_{\mathsf{R}}$*, outputs a witness* y *such that* $(Y, y) \in \mathsf{R}$*, or* $\perp$*.*

We note that in aSIG_2, the pVrfy algorithm enables public verifiability of the pre-signatures, e.g., aSIG_2 can be used in a three-party protocol where the third party needs to verify the validity of the generated pre-signatrue.

In the following, we formally define properties that a two-party adaptor signature scheme with aggregatable public keys aSIG_2 has to satisfy. These properties are similar to the ones for single party adaptor signature schemes. We start by defining two-party pre-signature correctness which, similarly to Def. 2 states that an honestly generated pre-signature and signature are valid, and it is possible to extract a valid witness from them.

Definition 11 (Two-Party Pre-Signature Correctness). *A two-party adaptor signature with aggregatable public keys* aSIG_2 *satisfies two-party pre-signature correctness, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, it holds that:*

$$\Pr\left[\begin{array}{c|l} \mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1 & pp \leftarrow \mathsf{Setup}(1^n), (sk_0, pk_0) \leftarrow \mathsf{Gen}(pp) \\ \wedge & (sk_1, pk_1) \leftarrow \mathsf{Gen}(pp), (Y, y) \leftarrow \mathsf{GenR}(1^n) \\ \mathsf{Vrfy}_{apk}(m; \sigma) = 1 & \widetilde{\sigma} \leftarrow \Pi_{\mathsf{pSign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m, Y) \\ \wedge & apk := \mathsf{KAg}(pk_0, pk_1) \\ (Y, y') \in \mathsf{R} & \sigma := \mathsf{Adapt}_{apk}(\widetilde{\sigma}, y), y' := \mathsf{Ext}_{apk}(\sigma, \widetilde{\sigma}, Y) \end{array}\right] = 1.$$

The unforgeability security definition is similar to Def. 9, except the adversary interacts with two oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ in order to generate signatures and pre-signatures, as in Def. 3. More precisely, in the $\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$ experiment defined below, $\mathcal{A}$ obtains access to *interactive*, stateful signing and pre-signing oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}$ and $\mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ respectively. Oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}$ and $\mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ simulate the honest party $\mathcal{P}_{1-b}$ during an execution of the protocols $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot\rangle}$ and $\Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot\rangle}$ respectively. Similar to Def. 9, both the protocols $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot\rangle}, \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot\rangle}$ employed by the respective oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ gets an oracle access to $\mathcal{A}$ as well.

Definition 12 (2-aEUF–CMA Security). *A two-party adaptor signature with aggregatable public keys* aSIG_2 *is unforgeable if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}(n)$ *is defined as follows:*

```
$\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_\mathrm{S}}, \mathcal{O}^b_{\Pi_\mathrm{pS}}}(pk_{1-b}, sk_b, pk_b)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(m^*, Y)$
7 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_\mathrm{S}}, \mathcal{O}^b_{\Pi_\mathrm{pS}}}(\widetilde{\sigma}, Y)$
8 : return $\left(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{\mathsf{KAg}(pk_0, pk_1)}(m^*; \sigma^*)\right)$

$\mathcal{O}^b_{\Pi_\mathrm{S}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m)$

$\mathcal{O}^b_{\Pi_\mathrm{pS}}(m, Y)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m, Y)$
```

The definition of two-party pre-signature adaptability follows Def. 4 closely. The only difference is that in this setting the pre-signature must be valid under the aggregated public keys.

Definition 13 (Two-Party Pre-Signature Adaptability). *A two-party adaptor signature scheme with aggregatable public keys* aSIG_2 *satisfies two-party pre-signature adaptability, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, statement and witness pairs* $(Y, y) \in \mathsf{R}$*, public keys* pk_0 *and* pk_1*, and pre-signatures* $\widetilde{\sigma} \in \{0,1\}^*$ *satisfying* $\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1$ *where* $apk := \mathsf{KAg}(pk_0, pk_1)$*, we have* $\Pr[\mathsf{Vrfy}_{apk}(m; \mathsf{Adapt}_{apk}(\widetilde{\sigma}, y)) = 1] = 1$.

Finally, we define two-party witness extractability.

Definition 14 (Two-Party Witness Extractability). *A two-party public key aggregatable adaptor signature scheme* aSIG_2 *is* witness extractable *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* ν *such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_2}$ *is defined as follows:*

```
$\mathsf{aWitExt}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_\mathrm{S}}, \mathcal{O}^b_{\Pi_\mathrm{pS}}}(pk_{1-b}, sk_b, pk_b)$
5 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(m^*, Y^*)$
6 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_\mathrm{S}}, \mathcal{O}^b_{\Pi_\mathrm{pS}}}(\widetilde{\sigma})$
7 : $apk := \mathsf{KAg}(pk_0, pk_1), y' := \mathsf{Ext}_{apk}(\sigma^*, \widetilde{\sigma}, Y^*)$
8 : return $(m^* \notin \mathcal{Q} \wedge (Y^*, y') \notin \mathsf{R} \wedge \mathsf{Vrfy}_{apk}(m^*; \sigma^*))$

$\mathcal{O}^b_{\Pi_\mathrm{S}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m)$

$\mathcal{O}^b_{\Pi_\mathrm{pS}}(m, Y)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m, Y)$
```

Note that the only difference between this experiment and the $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}$ experiment is that here the adversary is allowed to choose the statement/witness pair (Y^*, y^*) and that the winning condition additionally requires that for the extracted witness $y' \leftarrow \mathsf{Ext}_{apk}(\sigma^*, \widetilde{\sigma}, Y^*)$ it holds that $(Y^*, y') \notin \mathsf{R}$.

A two-party adaptor signature scheme with aggregatable public keys aSIG_2 is called *secure* if it satisfies the properties 2-aEUF–CMA security, *two-party pre-signature adaptability* and *two-party witness extractability.*

5.1 Generic transformation from $\mathsf{SIG}_2^{\mathsf{ID}}$ to $\mathsf{aSIG}_2^{\mathsf{ID,R}}$

We now present our generic transformation to achieve two-party adaptor signature schemes with aggregatable public keys from identification schemes. In its essence, this transformation is a combination of the transformations presented in Figs. 3 and 6. More precisely, similar to the transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{aSIG}^{\mathsf{ID,R}}$ presented in Fig. 3, we assume the existence of functions f_{shift}, f_{adapt} and f_{ext} with respect to the relation R. We then make use of the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol from the transformation in Fig. 6 to let parties agree on the randomness that is going to be used during the pre-signing process. However, unlike the transformation in Fig. 6, the resulting randomness is shifted by a statement Y for relation R using the function f_{shift}. The transformation can be found in Fig. 7.

```
$\Pi_{\mathsf{pSign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m, Y)$
1 : Parse $pk_i = ((1^n, pp_{\mathsf{C}}, \mathsf{crs}), pk_i'), i \in \{0,1\}$
2 : $(R_i, St_i, R_{1-i}) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_i, sk_{1-i}\rangle}(pp_{\mathsf{C}}, \mathsf{crs})$
3 : $R_{\mathsf{pre}} := f_{\mathsf{com\text{-}rand}}(R_0, R_1)$
4 : $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), h := \mathcal{H}(R_{\mathsf{sign}}, m)$
5 : $\tilde{s}_i \leftarrow \mathsf{P}_2(sk_i, R_i, h, St_i)$
6 : $\tilde{s}_{1-i} \leftarrow \Pi_{\mathsf{Exchange}}\langle \tilde{s}_i, \tilde{s}_{1-i}\rangle$
7 : $(h, \tilde{s}) := f_{\mathsf{com\text{-}sig}}(h, (\tilde{s}_i, \tilde{s}_{1-i}))$
8 : return $(h, \tilde{s})$
```

```
$\mathsf{pVrfy}_{apk}(m, Y; (h, \tilde{s}))$
1 : $\widehat{R}_{\mathsf{pre}} := \mathsf{V}_0(apk, h, \tilde{s})$
2 : return $h = \mathcal{H}(f_{\mathsf{shift}}(\widehat{R}_{\mathsf{pre}}, Y), m)$

$\mathsf{Adapt}_{pk}((h, \tilde{s}), y)$
1 : return $(h, f_{\mathsf{adapt}}(\tilde{s}, y))$

$\mathsf{Ext}_{pk}((h, s), (h, \tilde{s}), Y)$
1 : return $f_{\mathsf{ext}}(s, \tilde{s})$
```

Fig. 7: A two-party adaptor signature scheme with aggregatable public keys $\mathsf{aSIG}_2^{\mathsf{ID,R}}$ defined with respect to a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme and a hard relation R.

Theorem 4. *Assume that* $\mathsf{SIG}^{\mathsf{ID}}$ *is an* SUF–CMA*-secure signature scheme transformed using Fig. 2. Further, assume that* $f_{\mathsf{com\text{-}sig}}$, $f_{\mathsf{com\text{-}pk}}$ *and* $f_{\mathsf{dec\text{-}sig}}$ *satisfy the relation from Equations (4) and (5). Let* $f_{\mathsf{shift}}, f_{\mathsf{adapt}}$ *and* f_{ext} *be functions satisfying the relations from Equations (1) and (2), and* R *be a hard relation. Further, let* NIZK *be a non-interactive zero-knowledge proof system and let* C *be an extractable commitment scheme. Then the resulting* $\mathsf{aSIG}_2^{\mathsf{ID,R}}$ *scheme from the transformation in Fig. 7 is a secure two-party adaptor signature scheme with aggregatable public keys in the random oracle model.*

In order to prove Thm. 4, we must show that $\mathsf{aSIG}_2^{\mathsf{ID,R}}$ satisfies the *pre-signature correctness*, 2-aEUF–CMA *security*, *pre-signature adaptability* and *witness extractability* properties as described in Defs. 11 to 14 respectively. We provide the full proofs of the following lemmas in the full version of this paper [14] and only mention the intuition behind the proofs here. As mentioned in the introduction of this work, despite the fact that $\mathsf{aSIG}_2^{\mathsf{ID,R}}$ is constructed from $\mathsf{SIG}_2^{\mathsf{ID}}$, we require only $\mathsf{SIG}^{\mathsf{ID}}$ to be SUF–CMA-secure in order to prove 2-aEUF–CMA security for $\mathsf{aSIG}_2^{\mathsf{ID,R}}$.

Lemma 7 (Two-Party Pre-Signature Correctness). *Under the assumptions of Thm. 4,* $\mathsf{aSIG}_2^{\mathsf{ID,R}}$ *satisfies Def. 11.*

The proof of Lemma 7 follows directly from Equations (1) to (3) and the correctness of SIG_2 from Lemma 5.

Lemma 8 (2-$\mathsf{aEUF{-}CMA}$-Security). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 12.*

Proof Sketch: In a nutshell the proof of this lemma is a combination of the proofs of Lemmas 2 and 6, i.e., the proof is done by a reduction to the hardness of the relation R and the $\mathsf{SUF{-}CMA}$ of the underlying signature scheme. During the signing process, the challenger queries its $\mathsf{SUF{-}CMA}$ signing oracle and receives a signature σ. As in the proof of Lemma 6, the challenger programs the random oracle such that σ appears like a signature generated with the combined randomness of the challenger and the adversary. Simulating the pre-signing process is similar with the exception that before programming the random oracle, the randomness must be shifted using the function f_{shift}. Finally, the challenger and the adversary generate a pre-signature $\widetilde{\sigma}^* = (h, \tilde{s})$ on the challenge message m^* and the adversary outputs the forgery $\sigma^* = (h, s)$. If $f_{\mathsf{ext}}(s, \tilde{s})$ returns the y generated by the challenger, as in the proof of Lemma 2, the hardness of the relation R can be broken. Otherwise, using $f_{\mathsf{dec\text{-}sig}}$, it is possible to use the forgery provided by the adversary to extract a forgery for the $\mathsf{SUF{-}CMA}$ game.

Lemma 9 (Two-Party Pre-Signature Adaptability). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 13.*

Proof Sketch: The proof of Lemma 9 is analogous to the proof of Lemma 3.

Lemma 10 (Two-party Witness Extractability). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 14.*

Proof Sketch: The proof of Lemma 10 is very similar to the proof of Lemma 8 except that the adversary chooses Y now and thus, no reduction to the hardness of the relation R is needed.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) Emmy Noether Program *FA 1320/1-1*, by the *DFG CRC 1119 CROSSING* (project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain project* (grant nr. 16KIS0902) and by *NSCX project* (project number CS1920241NSCX008801) on *Design and Development of Blockchain based Technologies* in the *Department of Computer Science and Engineering, IIT Madras.*

References

[1] M. Abdalla et al. "Tighter Reductions for Forward-Secure Signature Schemes". In: *PKC 2013*. 2013.
[2] L. Aumayr et al. *Generalized Bitcoin-Compatible Channels*. Cryptology ePrint Archive, Report 2020/476. `https://eprint.iacr.org/2020/476.pdf`. 2020.
[3] M. Bellare and G. Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *ACM CCS 2006*. 2006.
[4] M. Bellare and S. Shoup. "Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles". In: *PKC 2007*. 2007.
[5] *Bitcoin Scripts*. `https://en.bitcoin.it/wiki/Script#Crypto`.
[6] *Bitcoin Wiki: Payment Channels*. `https://en.bitcoin.it/wiki/Payment_channels`.
[7] A. Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". In: *PKC 2003*. 2003.
[8] D. Boneh et al. "Compact Multi-signatures for Smaller Blockchains". In: *ASIACRYPT 2018, Part II*. 2018.

[9] D. Boneh et al. “Short Signatures from the Weil Pairing”. In: *ASIACRYPT 2001*. 2001.
[10] C. Decker and R. Wattenhofer. “A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels”. In: *Stabilization, Safety, and Security of Distributed Systems 2015*. 2015.
[11] A. Deshpande and M. Herlihy. “Privacy-Preserving Cross-Chain Atomic Swaps”. In: *FC 2020*. 2020.
[12] M. Drijvers et al. “On the Security of Two-Round Multi-Signatures”. In: *2019 IEEE Symposium on Security and Privacy*. 2019.
[13] L. Eckey et al. *Splitting Payments Locally While Routing Interdimensionally*. Cryptology ePrint Archive, Report 2020/555. https://eprint.iacr.org/2020/555. 2020.
[14] A. Erwig et al. *Two-Party Adaptor Signatures From Identification Schemes*. Cryptology ePrint Archive, Report 2021/150. https://eprint.iacr.org/2021/150. 2021.
[15] M. F. Esgin et al. “Post-Quantum Adaptor Signatures and Payment Channel Networks”. In: *ESORICS 2020*. 2020.
[16] M. Fischlin. “Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors”. In: *CRYPTO 2005*. 2005.
[17] L. Fournier. *One-Time Verifiably Encrypted Signatures A.K.A. Adaptor Signatures*. https://github.com/LLFourn/one-time-VES/blob/master/main.pdf. 2019.
[18] R. Gennaro et al. “Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security”. In: *ACNS 16*. 2016.
[19] S. Goldwasser and R. Ostrovsky. “Invariant signatures and non-interactive zero-knowledge proofs are equivalent”. In: *Annual International Cryptology Conference*. Springer. 1992.
[20] J. Groth et al. “Perfect Non-interactive Zero Knowledge for NP”. In: *EUROCRYPT 2006*. 2006.
[21] J. Gugger. *Bitcoin–Monero Cross-chain Atomic Swap*. Cryptology ePrint Archive, Report 2020/1126. https://eprint.iacr.org/2020/1126. 2020.
[22] L. C. Guillou and J.-J. Quisquater. “A “Paradoxical” Indentity-Based Signature Scheme Resulting from Zero-Knowledge”. In: *CRYPTO’88*. 1990.
[23] T. Hardjono and Y. Zheng. “A practical digital multisignature scheme based on discrete logarithms (extended abstract)”. In: *Advances in Cryptology — AUSCRYPT ’92*. 1993.
[24] J. Katz and N. Wang. “Efficiency Improvements for Signature Schemes with Tight Security Reductions”. In: *ACM CCS 2003*. 2003.
[25] E. Kiltz et al. “Optimal Security Proofs for Signatures from Identification Schemes”. In: *CRYPTO 2016, Part II*. 2016.
[26] D.-P. Le et al. *DDH-based Multisignatures with Public Key Aggregation*. Cryptology ePrint Archive, Report 2019/771. https://eprint.iacr.org/2019/771. 2019.
[27] S. Lu et al. “Sequential Aggregate Signatures and Multisignatures Without Random Oracles”. In: *EUROCRYPT 2006*. 2006.
[28] A. Lysyanskaya. “Unique signatures and verifiable random functions from the DH-DDH separation”. In: *Annual International Cryptology Conference*. Springer. 2002.
[29] G. Malavolta et al. “Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability”. In: *NDSS 2019*. 2019.
[30] G. Maxwell et al. “Simple Schnorr multi-signatures with applications to Bitcoin”. In: *Designs, Codes and Cryptography 2019* (2019).
[31] S. Micali et al. “Verifiable Random Functions”. In: *40th FOCS*. 1999.
[32] A. Miller et al. “Sprites and State Channels: Payment Networks that Go Faster Than Lightning”. In: *FC 2019*. 2019.
[33] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. http://bitcoin.org/bitcoin.pdf. 2009.
[34] K. Ohta and T. Okamoto. “A digital multisignature scheme based on the Fiat-Shamir scheme”. In: *Advances in Cryptology — ASIACRYPT ’91*. 1993.
[35] T. Okamoto. “A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems”. In: *ACM Trans. Comput. Syst.* 4 (1988).

[36] A. Poelstra. *Scriptless scripts*. https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf. 2017.

[37] J. Poon and T. Dryja. *The Bitcoin Lightning Network: Scalable Off-chain Instant Payments*. https://lightning.network/lightning-network-paper.pdf. 2016.

[38] R. L. Rivest et al. “How to Leak a Secret”. In: *ASIACRYPT 2001*. 2001.

[39] N. van Saberhagen. *CryptoNote v 2.0*. https://bytecoin.org/old/whitepaper.pdf.

[40] C.-P. Schnorr. “Efficient Signature Generation by Smart Cards”. In: *Journal of Cryptology* 3 (1991).

[41] S.-T. Shen et al. “Unique signature with short output from cdh assumption”. In: *International Conference on Provable Security*. Springer. 2015.

[42] E. Tairi et al. *A^2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs*. Cryptology ePrint Archive, Report 2019/589. https://eprint.iacr.org/2019/589. 2019.

[43] E. Tairi et al. *Post-Quantum Adaptor Signature for Privacy-Preserving Off-Chain Payments*. To appear at FC 2021. https://eprint.iacr.org/2020/1345.