Cryptography from Zero Knowledge

Advanced Security and New Constructions

Akira Takahashi

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

Cryptography from Zero Knowledge
Advanced Security and New Constructions

A Dissertation
Presented to the Faculty of Natural Sciences
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Akira Takahashi
June 28, 2022

Abstract

Zero knowledge interactive proof systems are a fundamental tool in modern cryptology, forming the foundation of many efficient cryptographic primitives, such as identification schemes and digital signatures. This thesis pushes forward the study of constructions based on zero knowledge proofs in four different directions: two in advanced security analysis of existing digital signatures, and two in new constructions supporting advanced functionalities.

Our first contribution lies in side-channel attacks on discrete log-based signature schemes, including ECDSA and Schnorr. We study the following extreme scenario of side-channel leakage: what if each signing attempt leaks less than 1 bit of information about internal, ephemeral randomness? We show that even such a mild leakage is exploitable in practice, by giving several theoretical improvements to an existing approach to solving the so-called hidden number problem (HNP).

In order to balance concerns of both randomness failures and the threat of fault attacks, some signature designs are advocating a "hedged" derivation of randomness. Our second contribution involves detailed security analyses of the fault resilience of hedged signatures constructed from three-round public-coin protocols in the random oracle model, using the methodology of provable security. As concrete case studies, we apply our results to hedged versions of EdDSA and Picnic signature schemes.

The third result comprises new distributed signing protocols in the random oracle model relying on standard lattice-based hardness assumptions. We study two similar classes of distributed signing, namely, n-out-of-n signature and multi-signature, both of which allow a group of signers to jointly produce a single signature on the same message. We realize relatively simple lattice-based two-round distributed signing, by carefully lifting several existing tricks known in the discrete log setting into the lattice world.

Our fourth contribution is in the domain of verifiable encryption. Verifiable encryption is a protocol where one can provide assurance that an encrypted plaintext satisfies certain properties or relations. In this part, we propose a novel, generic framework that realizes verifiable encryption protocols supporting a large class of relations and public-key encryption schemes, using zero-knowledge proof systems based on the MPC-in-the-head paradigm.

Resumé

Zero knowledge interaktive bevis-systemer er et fundamentalt værktøj for moderne kryptologi. Det er en grundsten for mange effektive kryptografiske primitiver, såsom identifikations protokoller og digitale signaturer. Denne afhandling bidrager til forskningen om zero knowledge protokoller på fire forskellige måder: to af disse vedrører sikkerhedsanalyser af eksisterende digitale signaturer og de resterende to vedrører nye konstruktioner der tillader avancerede funktionaliter i forhold til den krypterede eller signerede data.

Vores første bidrag til forskningsområdet vedrører side-channel angreb på signatur protokoller baseret på diskrete logaritmer, deriblandt både ECDSA og Schnorr. Vi kigger på det følgende eksempel på side-channel lækage: hvad hvis hvert signerings forsøg lækker mindre end 1 bit af information omkring den interne tilfældighed? Vi viser at selv sådan en lille lækage kan udnyttes i praksis, ved at bruge informationen fra denne lækage til at forbedre en eksisterende måde at løse det såkaldte Hidden Number Problem (HNP).

I forsøget på at balancere frygten for brugen af en "dårlig" tilfældighed og specifikke fejl der kan opstå ved brugen af en mere deterministik tilfældighed, så er nogle signatur protokoller begyndt at bruge en mere lagdelt metode til at tilføre tilfældighed til protokollen. Vores andet bidrag involverer en detaljeret sikkerhedsanalyse af disse protokollers evne til at modstå fejl ved denne beregning af tilfældighed, når de er konstrueret ud fra en tre-runde public-coin protokol i modellen med tilfældigheds orakler. Som konkrete tilfælde, udfører vi vores analyse på lagdelte versioner af EdDSA og Picnic signatur protokoller.

Vores tredje resultat omfatter nye distribuerede signerings protokoller i modellen der tillader tilfældigheds orakler. Disse protokoller kræver kun at standard lattice-baserede problemer er svære at løse. Vi kigger på to lignende klasser af distribuerede signerings protokoller, specifikt dem der producerer en n-ud-af-n signatur, og multi-signaturer. Begge disse klasser tillader en gruppe at producere én enkelt signatur for én besked, sammen. Vi konstruerer en relativt simpel lattice-baseret distribueret-signerings protokol der kun bruger to runder, ved at påpasseligt bruge forskellige eksisterende tricks man normalt kan bruge når man arbejder med diskrete logaritmer, i en lattice-baseret model i stedet.

Vores sidste bidrag er ift verificerbar kryptering. Verificerbar kryptering er en protokol hvor man kan forsikre folk om at en krypteret tekst tilfredsstiller forskellige egenskaber, eller relationer. Til denne del giver vi en ny, generisk opskrift der kan bruges til at opnå verificerbare krypterings protokoller, der supporterer en stor gruppe af relationer og public-key krypterings metoder, ved at bruge zero-knowledge bevis systemer baseret på MPC-in-the-Head paradigmet.

Acknowledgments

This dissertation wouldn't exist without the support of a number of people. I am grateful to my wonderful advisors, Claudio Orlandi and Diego de Freitas Aranha, who guided me through my journey towards a PhD. Claudio has always led by example: his enthusiasm for research, innovative mind, and caring for every individual in the group have been a huge inspiration to me. I have learned a great deal from Diego's inclusiveness, dedication to the research community, and passion for addressing real-world problems using technology. Thank you, Claudio and Diego, for constantly caring about what is best for my life and career, teaching me the importance of a professional mindset, and everything in between, not to mention the countless technical discussions we had.

I have been lucky enough to be mentored by multiple other people, whether officially or unofficially. Mehdi Tibouchi, my master's advisor at Kyoto University and internship mentor at NTT, has generously supported me in many collaborative projects. I am deeply indebted to Mehdi for always enlightening me with fresh ideas and sharing his profound knowledge of restaurants in Tokyo. Greg Zaverucha, my internship mentor at Microsoft Research, has continuously assisted me since the beginning of my PhD. I would like to thank Greg for always giving me hands-on guidance and making my summer in Redmond joyful despite the pandemic. Chaya Ganesh has been leading exciting collaborative projects even after she left for India. Thank you, Chaya, for nurturing my interest in zero knowledge and always being open to discussions regardless of the time difference. From Ivan Damgård I've learned a considerable amount both through our project and his lectures. I owe a lot to his direction and extremely deep understanding of cryptology. Stefano Tessaro kindly agreed to host me as a visitor at the University of Washington. I would like to thank him for involving me in meetings and for exchanging ideas. Thanks to Masayuki Abe and Tatsuaki Okamoto, who were my master's co-advisors, I was introduced to this amazing community. Without meeting them none of this would have happened.

I am indebted to every single person who co-authored papers with me. Thank you, Emil Madsen Bennedsen, Sebastian Berndt, Cecilia Boschini, Matteo Campanelli, Thomas Eisenbarth, Thomas Espitau, Pierre-Alain Fouque, François Gérard, Felipe Rodrigues Novaes, Mahak Rakesh Pancholi, Mélissa Rossi, Okan Seker, Daniel Tschudi, Alexandre Wallet, Luca Wilke, Yuval Yarom, and Yang Yu. Every single bit of collaboration has been precious to me.

Thanks to my office mate Alexander Munch-Hansen this thesis has been completed with a proper Danish abstract. I am grateful to everyone in the Aarhus crypto group for creating a lively and enjoyable environment to work in. Sporadic conversations happening at the crypto couch were always refreshing. I would also like to thank everyone at NTT Musashino R&D Center for welcoming my random visits. I am thankful to Dario Fiore and Nadia Heninger for being part of my committee and taking the time to read this thesis.

Special thanks to my parents, Akihiko and Shiori Takahashi, for their unconditional love and support. Last but definitely not least, I would like to express my sincere gratitude to my partner Mirka Niemi for believing in me and for always being there during difficult times.

Akira Takahashi,
Aarhus, June 28, 2022.

To Koji Takahashi

Contents

Abstract i

Resumé iii

Acknowledgments v

Contents vii

1 Introduction 1
- 1.1 Proofs and Identifications . . . 1
- 1.2 Removing Interactions . . . 6
- 1.3 Overview of Thesis . . . 9
- 1.4 Other Publications . . . 15

I Advanced Security Analysis 17

2 *LadderLeak* 19
- 2.1 Introduction . . . 19
- 2.2 Preliminaries . . . 22
- 2.3 Timing Attacks on Montgomery Ladder . . . 27
- 2.4 Improved Analysis of Bleichenbacher's Attack . . . 30
- 2.5 Experimental Results . . . 37
- 2.6 Software Countermeasures . . . 40

3 Security of Hedged Fiat-Shamir Signatures 41
- 3.1 Introduction . . . 41
- 3.2 Preliminaries . . . 46
- 3.3 Formal Treatment of Hedged Signatures . . . 50
- 3.4 Security of Hedged Signatures Against Fault Attacks . . . 54
- 3.5 Analysis of XEdDSA . . . 62
- 3.6 Analysis of Picnic2 . . . 62
- 3.7 Concluding Remarks . . . 64

II New Constructions 65

4 Two-Round Multi-Party Signing from Lattices 67

4.1 Introduction 67
4.2 Preliminaries 76
4.3 DS_2: Two-round n-out-of-n Signing from Module-LWE and Module-SIS . . 85
4.4 MS_2: Two-round Multi-signature in the Plain Public Key Model 98
4.5 Lattice-Based Commitments 100

5 Verifiable Encryption from MPC-in-the-Head 105
5.1 Introduction 105
5.2 Preliminaries 113
5.3 Our Transform 117
5.4 Methods for Compressing Ciphertexts 123
5.5 Concrete Instantiations 127
5.6 Conclusion and Future Work 131

Bibliography 133

Chapter 1

Introduction

1.1 Proofs and Identifications

1.1.1 Interactive Proofs

Little Victor is trying to solve a math homework problem that must be answered with a "Yes" or "No". After some trial and error, he feels the problem is way above his head, so he decides to ask for help from his brilliant classmate Peggy, and in return agrees to give her some candies in case she solves it. Peggy gets back to Victor after a few days, claiming the answer is "Yes". Before handing in his homework to school, Victor needs to validate her claim and would also like to learn the detailed step-by-step solution to the problem. However, from her experience Peggy knows Victor to be a fickle-minded person, so she is afraid that Victor will change his mind and not give her the candies after she teaches him the details. Is there any good way for Peggy to convince Victor that the answer is indeed "Yes", without revealing how she came to this conclusion?

An *interactive proof system*, first formalized as a concept by Goldwasser, Micali, and Rackoff [GMR85, GMR89], offers a solution to these kinds of scenarios. Let us abstract out the problem. Consider a *language* L, a set of problem instances to which the answer is "Yes". Peggy and Victor are now modeled as *probabilistic Turing machines*, *prover* $\mathcal{P}$ and *verifier* $\mathcal{V}$, respectively. Typically, $\mathcal{P}$ has *unbounded* computational power so that she can decide whether a *statement* $x \in L$ is true or not, whereas $\mathcal{V}$ should run in *polynomial time* in the length of x. The goal of the proof system is that $\mathcal{P}$ convinces $\mathcal{V}$ that x is indeed in L after going through some interactive process specified in a *protocol*.

The common theme underlying this thesis is cryptographic constructions that arise from *public-coin* protocols, a special case of general interactive proof systems. The concept was first coined by Babai's work [Bab85], wherein it is formulated as the *Arthur-Merlin* game using interaction between mythical figures as a metaphor. A public-coin protocol proceeds as follows: the prover $\mathcal{P}$ sends to the verifier $\mathcal{V}$ the first message a_1, $\mathcal{V}$ responds with *challenge* e_1 sampled uniformly at random from a set of bit strings Ch_1 (i.e., $\mathcal{V}$ flips random coins publicly), $\mathcal{P}$ sends the second message a_2, which is responded with $e_2 \in \mathsf{Ch}_2$ by $\mathcal{V}$, and so forth. The vast majority of existing protocols for concrete languages, such as quadratic residues [GMR85], graph isomorphism [GMW86], graph 3-colorability [GMW86], and Hamiltonian graphs [Blu86], indeed follow this blueprint, whereas some rely on the verifier's ability to keep their coins *private*, e.g., quadratic non-residues [GMR85] and graph non-isomorphism [GMW86] (although even they can be turned into public-coin protocols

by adding extra rounds [GS86]).

While one can consider protocols with an arbitrary number of rounds, the *canonical* form of public-coin protocols has three rounds of interaction:

1. $\mathcal{P}$ sends a *commit* message a.
2. $\mathcal{V}$ returns a *challenge* $e \in \mathsf{Ch}$.
3. $\mathcal{P}$ *responds* with z.

Most constructions we will look at in later chapters follow this simple format.

Soundness: protecting against dishonest provers Clearly, one should at least design a protocol satisfying *completeness*, meaning that if both parties correctly follow the protocol specification on input $x \in L$, $\mathcal{V}$ should output 1, indicating "I am convinced". Suppose we have designed a protocol satisfying completeness. However, we are not done yet, because we still have to consider a situation where either party *cheats* during the protocol. As you may have noticed, not everyone behaves honestly in this world, so we need to devise a mechanism to catch or prevent dishonesty—this is exactly why we study cryptology. So what if the statement x is *not* in the language but $\mathcal{P}$ tricks $\mathcal{V}$ into believing $x \in L$? This is obviously bad in the initial scenario, because it implies that Peggy can get away without really doing Victor's homework, yet she gets free candies! To prevent such a situation, proof systems should have *soundness*, meaning that any potentially cheating prover $\mathcal{P}^*$ has a hard time convincing $\mathcal{V}$ if $x \notin L$. Or in real life, it would be probably reasonable to assume cheating provers only have bounded computational resources. If soundness only holds for polynomial-time $\mathcal{P}^*$, a system is said to be an *argument* instead of proof.

Zero Knowledge: protecting against curious verifiers There can be potentially many protocols realizing a secure proof system for a language L. For the sake of simplicity, let us consider a language L in the class NP (although it is known that a class of languages for which interactive proofs exist is as large as PSPACE [LFKN90, Sha90]). This implies that there exists the corresponding *binary relation* $R_L := \{(x, w) : x \in L \wedge w \text{ is an NP-witness for } x\}$. An obvious way for $\mathcal{P}$ to "prove" $x \in L$ is to run the following protocol: she sends a *witness* w explaining $x \in L$, and $\mathcal{V}$ verifies w by checking $(x, w) \in R_L$, which can be indeed carried out in polynomial time. This naïve approach, however, is not really a satisfactory solution whenever confidentiality matters. Notice that even though w can be efficiently tested, the prover is giving out far more than what's necessary to convey just 1-bit of information, i.e., $x \in L$. In certain situations, such as the above motivating example, $\mathcal{P}$ may not want to disclose anything else than the fact that $x \in L$.

Zero knowledge (ZK) is an additional important property for proof systems. On a high-level, it requires that a verifier should gain from the protocol execution nothing more than the fact that $x \in L$, even if he behaves dishonestly. Perhaps surprisingly, mathematically defining what it means to "learn nothing" turned out to be an intricate task. *Simulation paradigm* is one of the most important concepts in cryptography to deal with such an issue, and has seen success in defining security notions for a variety of cryptographic objects, such as zero knowledge proofs, multi-party computation, and public key encryption [Gol01, HL10, CDN15, Lin17b]. We say a proof system is (computationally) zero knowledge if for any probabilistic polynomial-time cheating verifier $\mathcal{V}^*$ there exists an efficient simulator $\mathcal{S}$ that can produce the *view* of $\mathcal{V}^*$ (i.e. all the information that $\mathcal{V}^*$ can observe during an execution of the protocol). The rationale behind this somewhat quirky

formulation becomes clear from *real vs ideal* reasoning. Let us first consider the *ideal world*, where there is no prover, and only $\mathcal{V}^*$ with input statement x exists. A simulator $\mathcal{S}$ can be seen as an imaginary entity spawned in the ideal world, and thus whatever $\mathcal{S}$ can produce essentially corresponds to the kind of information that an attacker $\mathcal{V}^*$ can compute on his own in this ideal world. We then compare this to what $\mathcal{V}^*$ sees in the *real world*, where he runs an actual protocol with $\mathcal{P}$. If the two worlds are *indistinguishable* with each other, we can conclude that $\mathcal{V}^*$ has "learned nothing new" in the real world.

Designing a protocol meeting full-fledged zero knowledge is often challenging, so it is also useful to relax the requirement by considering *honest verifier zero knowledge (HVZK)*. Here, we assume that a verifier $\mathcal{V}^*$ does not deviate from the protocol while it may try to retrieve some useful information by interacting with a prover $\mathcal{P}$. As we shall see later, HVZK turns out sufficient in many cases for constructing other cryptographic primitives from proof systems.

Proof of Knowledge: making sure the prover *knows* something Finally, we observe that there are situations where just plain soundness is not an interesting property. Suppose a protocol for the language consisting of group elements for which a *discrete logarithm* exists:

$$L_{\mathrm{DL}} := \left\{ x \ : \ \exists w \in [0, q-1] : g^w = x \bmod p \right\}$$

where p and q are assumed to be primes such that $q|p-1$ and $g \in \mathbb{Z}_p^*$ generates the cyclic subgroup of order q. In fact, achieving soundness for such a language is trivial since $\langle g \rangle$ is the only subgroup of order q in $\mathbb{Z}_p^*$ and any verifier can easily check whether x belongs to $\langle g \rangle$ by testing $x^q = 1 \bmod p$. In other words, it is rather pointless to consider *proof of membership* for languages like L_{DL}. What one should strive for instead is *knowledge soundness*: at a high-level, it requires that the prover actually *knows* a right witness w corresponding to the statement x whenever the verifier is convinced. If a proof system Π satisfies both completeness and knowledge soundness, we say Π is *proof of knowledge (PoK)*. This stronger version of soundness already appeared in [GMR85] as a concept and later developed into formal definitions in subsequent works [FFS88, BG93], where knowledge soundness is defined with respect to the existence of an efficient *extractor* that outputs a valid witness using any successful prover as a subroutine. The most typical approach to proving knowledge soundness is via *rewinding*. Here, we construct an extractor $\mathcal{E}$ that has oracle access to the prover $\mathcal{P}^*$ and eventually extracts a witness w by resetting $\mathcal{P}^*$'s state multiple times, and by feeding different verifier messages to $\mathcal{P}^*$ whenever reset. For canonical three-round protocols, it is also useful to define a closely related notion called *special soundness*: a valid witness for x can be computed given two accepting transcripts sharing the first prover message i.e., (a, e, z) and (a, e', z'). Assuming special soundness, one can construct such an extractor for knowledge soundness that halts in expected polynomial time [CDS94, Dam10].

Taken together, Cramer in his PhD thesis [Cra96] introduced Σ-*protocols*, characterizing canonical three-round protocols with special soundness and HVZK. As we shall see soon, Σ-protocols form the foundation of various efficient cryptographic protocols/primitives and have been extensively used in the literature to date.

Properties in a nutshell All in all, we can summarize the following standard properties of interactive proof systems.

- **Completeness** If both parties honestly follow the protocol specifications on input $x \in L$, then $\mathcal{V}$ outputs 1.

- **Soundness** For sufficiently large false statement $x \notin L$, for any potentially malicious prover $\mathcal{P}^*$, $\mathcal{V}$ after interacting with $\mathcal{P}^*$ on input x outputs 1 except with probability negligible in the length of x.
- **Special Soundness** (for three-round protocols) Given two transcripts (a, e, z) and (a, e', z') such that $e \neq e'$, if they both verify with respect to x, one can efficiently extract a witness w such that $(x, w) \in R_L$.
- **Knowledge Soundness** There exists an efficient extractor $\mathcal{E}$ satisfying the following requirement: for any true statement $x \in L$, for any potentially malicious prover $\mathcal{P}^*$, if $\mathcal{V}$ after interacting with $\mathcal{P}^*$ on input x outputs 1, $\mathcal{E}$ having access to $\mathcal{P}^*$ outputs w such that $(x, w) \in R_L$.
- **Zero Knowledge** For sufficiently large true statement $x \in L$, for any potentially malicious verifier $\mathcal{V}^*$, there exists an efficient simulator $\mathcal{S}$, that outputs a string indistinguishable with the view of $\mathcal{V}^*$ while interacting with $\mathcal{P}$ on input x.
- **Honest Verifer Zero Knowledge** This is a weaker version of zero knowledge. The difference is that the simulator only needs to simulate the view of *honest* verifier $\mathcal{V}$ correctly following the protocol.

Note that the above definitions are rather informal since we have not defined precise syntax of protocol execution, what it means by "efficient", flavors of indistinguishability (i.e. perfect, statistical, or computational), etc. Later chapters will contain more formal definitions suitable for each context.

1.1.2 Identification Schemes

An *identification scheme* is a closely related object to proof systems. A scheme ID is a tuple of three algorithms $(\mathsf{IGen}, \mathsf{P}, \mathsf{V})$, where IGen is an *instance generator* returning a key pair $(\mathsf{pk}, \mathsf{sk})$ on security parameter as input, and P and V are analogous to prover and verifier of proof systems. A prover P first obtains a key pair $(\mathsf{pk}, \mathsf{sk})$ generated by IGen and keeps sk secret while publishing pk in public, so that any verifier V has access to pk. Now the goal of identification is 1) that V makes sure the person he is talking to is the genuine owner of sk paired with pk, and 2) that P successfully convinces V without leaking her secret identity sk. One may notice that these requirements are very analogous to knowledge soundness and zero knowledge of proof systems, respectively. Taking advantage of this observation, Fiat and Shamir [FS87] and Feige, Fiat, and Shamir [FFS88] constructed the first ZKPoK-based identification schemes. Since then many efficient identifications have been proposed in the literature, such as the RSA-based one by Guillou and Quisquarter [GQ88] and the discrete log-based one by Schnorr [Sch90, Sch91]. To see how proof and identification are related, let us first review security notions of identification schemes, following the formulations of Abdalla-An-Bellare-Namprempre [AABN02] and Kiltz-Masny-Pan [KMP16]. Consider the following security game between a cheating prover P^* and a challenger C.

1. The challenger C runs an instance generator as $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{IGen}(1^\kappa)$ and hands pk over to the adversary P^*
2. The challenger C returns polynomially many *transcripts* (i.e., the messages exchanged between honest $\mathsf{P}(\mathsf{sk}, \mathsf{pk})$ and $\mathsf{V}(\mathsf{pk})$ during protocol executions) to the adversary P^*.
3. The adversary P^* initiates an identification protocol with an honest verifier $\mathsf{V}(\mathsf{pk})$.
4. P^* wins if and only if V outputs 1 after the protocol execution.

The step highlighted in orange is optional. Without that step (resp. with that step), if the probability that any polynomial time adversary P^* wins the above game is negligible in κ, we say ID is $\mathsf{IMP\text{-}KOA}$-secure, i.e., secure against *impersonation under key only attack* (resp. $\mathsf{IMP\text{-}PA}$-secure, i.e., secure against *impersonation under passive attack*).

So intuitively, once a Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is given, it seems that we immediately obtain a secure identification scheme by simply defining $\mathsf{pk} := x$, $\mathsf{sk} := w$, $\mathsf{P} := \mathcal{P}$ and $\mathsf{V} := \mathcal{V}$. But what about IGen? Notice that security requirements for proof systems say nothing about the hardness of relation. This means one can in principle construct a proof system which is provably HVZK and special sound, yet recovering a witness from the statement is easy: e.g., consider Schnorr's Σ-protocol for a "weak" discrete log language consisting of points on vulnerable elliptic curves. In this case, whatever IGen outputs, anyone can compute a supposedly secret identity sk from pk, making the corresponding identification scheme completely useless in practice![1] To circumvent this, we must assume the existence of a *hard* instance generator IGen for relation R_L, meaning that IGen samples a pair $(\mathsf{pk}, \mathsf{sk})$ from R_L such that it is computationally hard to recover sk given only pk. Now, it is easy to see Σ-protocols give rise to secure identification schemes. To provide a brief background on common patterns of security reduction in this thesis, we will also give a sketch of proof.

Theorem 1.1 (Secure identification from Σ-protocol (informal))**.** *Let L be an NP-language and R_L be the associated NP-relation. Suppose $\Pi = (\mathcal{P}, \mathcal{V})$ is a Σ-protocol for R_L, that is, Π is complete, special sound, and HVZK. If IGen is a hard instance generator for R_L, then the corresponding $\mathsf{ID} = (\mathsf{IGen}, \mathsf{P}, \mathsf{V})$ is $\mathsf{IMP\text{-}PA}$-secure.*

Proof sketch. Let us first prove $\mathsf{IMP\text{-}KOA}$ only using special soundness. Given a cheating prover P^* winning the $\mathsf{IMP\text{-}KOA}$ game, consider the following reduction $\mathcal{A}$ breaking the hardness of IGen:

1. Upon receiving an instance pk from IGen, the reduction forwards pk to P^*.
2. Upon receiving the first protocol message a from P^*, $\mathcal{A}$ plays an honest verifier by returning uniformly random challenge e from Ch.
3. If P^* successfully convinces $\mathcal{A}$ by returning a response z, $\mathcal{A}$ resets P^*'s state to the point where P^* outputs the first message a, and then answers with freshly sampled challenge $e' \in \mathsf{Ch}$. Otherwise, $\mathcal{A}$ halts with output $\perp$.
4. Upon receiving a response z' from P^*, if the second run of the protocol also convinces $\mathcal{A}$ and $e \neq e'$, then $\mathcal{A}$ outputs a witness $w = \mathsf{sk}$ extracted from two transcripts (a, e, z) and (a, e', z') that are accepting with respect to pk (Note this is guaranteed thanks to special soundness). Otherwise, $\mathcal{A}$ halts with output $\perp$.

One can show the above strategy successfully returns a valid witness with probability at least $(\mathsf{acc} - 1/|\mathsf{Ch}|)^2$ thanks to the *reset lemma* [BP02, KMP16], where acc is the probability that P^* wins the $\mathsf{IMP\text{-}KOA}$ game in a single run of the protocol.

What about $\mathsf{IMP\text{-}PA}$? Notice the reduction $\mathcal{A}$ cannot return transcripts by simply running actual P and V, because this requires the knowledge of sk, which $\mathcal{A}$ does not know. This is where HVZK of the underlying Σ-protocol kicks in: now between Step 1 and 2, $\mathcal{A}$ generates a bunch of *simulated* transcripts by running the HVZK simulator $\mathcal{S}$ on input

[1] Note that this still does not contradict the requirement of ZK, since if attackers can in any case compute sk from pk alone, they have learned "nothing new" from the protocol executions.

pk and passes them to P^*. Thanks to the HVZK property, cheating provers have a hard time distinguishing these simulated transcripts from real ones, and one can still break the hardness of IGen using the standard hybrid arguments. □

We close this section by noting other approaches towards secure identifications that do not necessarily rely on HVZK and special soundness. First, one could exploit *witness indistinguishability (WI)* [FS90] instead of HVZK to meet $\mathsf{IMP\text{-}PA}$. The property roughly states that any verifiers have a hard time guessing which of potentially many witnesses are used by the prover. Hence, a challenger C can simply generate honest transcripts with one of many witnesses, while the witness extracted from a cheating prover P^* differs from the one used by C with high probability, which helps security reduction break the hardness of instance generator. Several ways to realize WI appeared in the literature, e.g., OR-composition of Σ-protocols [CDS94, CPS+16a, CPS+16b, FHJ20], the discrete log-based protocol by Okamoto [Oka93], and lattice-based protocols by Lyubashevsky [Lyu08, Lyu09]. Second, it is also possible to rely on the hardness of deciding membership for a language instead of special soundness. This approach assumes the existence of *lossy key generator* LGen, which outputs some fake key pk' for which no associated sk exists (i.e., $\mathsf{pk}' \notin L$) with high probability, while it is hard for an adversary to distinguish such a fake key from $\mathsf{pk} \in L$ generated via the legitimate generator IGen. Given this, a challenger C can cheat by running LGen instead of IGen without P^* noticing. Now it should be infeasible for P^* to convince the verifier since no valid secret identity exists to begin with. This elegant approach is known as *lossy identifications* in the literature [KW03, GJKW07, AFLT16, KLP17, KLS18] and helps us avoid subtle reduction loss caused by rewinding.

1.2 Removing Interactions

1.2.1 The Fiat-Shamir Paradigm

The protocols we have seen so far require some interaction between prover and verifier. In practice, however, it would be ideal to avoid interaction so that both parties do not have to be online at the same time. Interaction also causes subtle issues related to *parallel repetitions*: if the size of challenge space is limited, one may want to run in parallel many instances of the same protocol to guarantee sufficiently small soundness error, e.g., $< 2^{-128}$. Although HVZK does hold under parallel repetitions, it is not necessarily the case for full-fledged ZK [GK96]. *Non-interactive* proofs and identifications are attractive protocols from both practical and theoretical standpoints. The seminal work of Fiat and Shamir [FS87] offered a generic conversion method turning public-coin interactive proofs/identifications non-interactive, which is often referred to as the *Fiat-Shamir transform* in the literature. On a high-level, a non-interactive prover obtained by the Fiat-Shamir transform plays an honest verifier by hashing the first prover message together with a statement[2], to derive challenges from the hash value on her own. It turns out that this simple and neat idea has more applications than just removing interactions.

[2]It is worth mentioning that including a statement is crucial to retaining the security of the resulting non-interactive *proofs*. In fact, Bernhard, Pereira, and Warinschi [BPW12] presented devastating attacks completely breaking the soundness of Fiat-Shamir NIZK if the statement is missing in the hash input. Interestingly, some Fiat-Shamir *signatures* on the other hand retain their security without hashing the statement (i.e., the public key) even in the multi-user setting [KMP16].

1.2.2 Digital Signature from Identification

The Fiat-Shamir transform is also one of the most popular approaches to constructing *digital signatures* [DH76]. A cryptographic signature scheme, denoted by a tuple of three algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, can be thought of as a digital analogue of real-life handwritten signatures. The *signer* first generates a key pair $(\mathsf{sk}, \mathsf{pk})$ by running the key generation algorithm Gen, where pk roughly corresponds to the "style" of his/her handwritten signature recorded in a public registry, and sk is essentially the secret "muscle memory" enabling one to reproduce that particular style. On receiving a *message* m to be signed, the signer runs the Sign algorithm on input (sk, m), returning a *signature* on m. Then anyone in the world knowing pk should be able to *verify* a pair (σ, m) using the algorithm Ver, as long as it was produced by the legit signer.

What about security? Goldwasser, Micali, and Rivest [GMR88] formalized different security notions for digital signatures, depending on the combination of attack behaviors and the definition of "forgery". Here we recall two of them forming the de facto standards nowadays. The bare minimum requirement is that nobody except for the actual owner of sk should be able to create a valid signature on *any* message, given just pk as input. This is the so-called $\mathsf{UF\text{-}KOA}$ security (*existential unforgeablility under key only attack*). This however is not sufficient for modeling real-world adversarial behaviors, because malicious attackers may adaptively ask the owner of sk to sign many messages of their own choice and later try forging a signature on new message m^* after observing these legitimate signatures.In this setting, the attacker's strategy might vary, e.g., they might manage to forge by carefully combining the existing signatures, or they might even completely recover sk if the Sign algorithm leaks partial information about sk per each query. The $\mathsf{UF\text{-}CMA}$ (*existential unforgeablility under (adaptive) chosen message attack*) is a stronger security guarantee to rule out such attacks. Put formally, a signature scheme is said to be $\mathsf{UF\text{-}CMA}$ secure if for any probabilistic polynomial time forger F, the probability that F wins the following game is negligible in security parameter κ.

1. The challenger C runs a key generation algorithm as $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$ and hands pk over to the forger F.
2. On receiving a query m from F, the challenger C runs $\mathsf{Sign}(\mathsf{sk}, m)$ to obtain σ and returns σ to F.
3. The forger F outputs a message-forgery pair (m^*, σ^*).
4. F wins if and only if Ver outputs 1 on input $(\mathsf{pk}, m^*, \sigma^*)$ *and* m^* has never been queried to the challenger.

The step highlighted in orange is optional. Without that step, it degrades to the $\mathsf{UF\text{-}KOA}$ game. The reader may notice similarities with the security notions for identification schemes we discussed above: the chosen message attack queries correspond to passive attacks of the $\mathsf{IMP\text{-}PA}$ game, and the winning condition of a signature forger is somewhat analogous to that of $\mathsf{IMP\text{-}KOA}$/$\mathsf{IMP\text{-}PA}$. Indeed, they are closely related to each other as the Fiat-Shamir turns canonical identification schemes into signature schemes. Given $\mathsf{ID} = (\mathsf{IGen}, \mathsf{P}, \mathsf{V})$ and some cryptographic hash function $\mathsf{H} : \{0,1\}^* \to \mathsf{Ch}$, the Fiat-Shamir-transformed signature scheme $\mathbf{FS}[\mathsf{ID}, \mathsf{H}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is defined as follows.

- $\mathsf{Gen}(1^\kappa)$ is identical to $\mathsf{IGen}(1^\kappa)$
- $\mathsf{Sign}(\mathsf{sk}, m)$ proceeds as follows

1. Run the first stage of P on input sk to produce a.
2. Derive challenge $e = \mathsf{H}(\mathsf{pk}, a, m)$.
3. Run the second stage of P to obtain z.
4. Output $\sigma := (a, z)$ as a signature on m.

- $\mathsf{Ver}(\mathsf{pk}, \sigma', m)$ parses σ' as (a', z'), and outputs 1 if and only if V of the underling canonical identification scheme accepts $(\mathsf{pk}, a', e', z')$, where e' is recovered by computing hash $\mathsf{H}(\mathsf{pk}, a', m)$.

1.2.3 Provable Security of Fiat-Shamir

Many modern, efficient signature schemes, such as the Schnorr signature [Sch90], follow this format, and Fiat-Shamir based signatures have been successfully deployed in today's real-world systems. But what about provable security? Can we jump into the conclusion that $\mathbf{FS}[\mathsf{ID}, \mathsf{H}]$ is a secure signature if H is instantiated with standard cryptographic hash functions, such as SHA-256? Things are not quite so simple in the *standard model* because several counterexamples exist in the literature [GK03, BDG+13]. In particular, Goldwasser and Kalai [GK03] proved the existence of secure canonical identification schemes ID for which $\mathbf{FS}[\mathsf{ID}, \mathsf{H}]$ yields universally fogeable digital signature schemes for *any* hash function H. This negative result indicates that the Fiat-Shamir paradigm *fails* for some schemes and that one has to carefully investigate the properties needed for the function H by looking at inner workings of each specific ID, leading to a recent line of research striving for secure instantiation of Fiat-Shamir in the standard model, e.g., [CCRR18, CCH+19].

So despite its simplicity, provable security of the Fiat-Shamir paradigm turned out to be extremely subtle. Luckily, there is a somewhat heuristic, yet widely accepted methodology that can make our life much easier: the *random oracle model (ROM)*, originally introduced by Bellare and Rogaway [BR93], is a powerful paradigm that attempts to give proof of security by assuming the existence of a "random oracle" $\mathcal{O}$. The oracle $\mathcal{O}$ essentially behaves like an ideal hash function, in that 1) anyone in the world has black-box access to $\mathcal{O}$, 2) $\mathcal{O}$ returns a uniformly random bit string of particular length on each distinct input, and 3) $\mathcal{O}$ keeps track of all previous input-output pairs and consistently returns the same output associated with a particular input if the input was previously queried by someone.

Clearly, such a convenient oracle does not exist in real life and there are some (albeit contrived) schemes that are proven secure in the ROM, yet become insecure once the RO is instantiated with any concrete hash function [CGH98, GK03]. Arguably, our community seems to have gained some confidence in relying on proofs in the ROM, perhaps because we have never observed "real-world" schemes that turned out insecure due to the RO [KM15]. The recent surge of the "quantum" random oracle model (QROM) [BDF+11] (where the adversary is modeled as a quantum computer and is allowed to query the RO with *quantum superposition*) in the literature also indicates that the overall methodology is considered sound, while it greatly simplifies security proof. Admitting the usefulness of heuristics, previous works analyzed security of the Fiat-Shamir transform in the classical ROM [OO98, PS00a, AABN02, FKMV12, KMP16] and in the quantum ROM [DFG13, Unr17, KLS18, LZ19, DFMS19, DFM20]. The RO methodology has further succeeded in enabling *online extractability* of some schemes [Pas03, Fis05, Unr15, Kat21, DFMS21, DFMS22], circumventing the necessity of rewinding the adversary when proving knowledge soundness.

We are now set out to recall security of Fiat-Shamir signatures in the ROM, following the result of [AABN02].

Theorem 1.2 (Secure signature from identification in the ROM (informal))**.** *Let* $\mathsf{ID} = (\mathsf{IGen}, \mathsf{P}, \mathsf{V})$ *be an* IMP-PA*-secure canonical three-round identification scheme. If* H *is modeled as a random oracle, then* $\mathbf{FS}[\mathsf{ID}, \mathsf{H}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is a* UF-CMA*-secure signature scheme in the random oracle model.*

Proof sketch. Let us first prove UF-KOA only assuming IMP-KOA of ID. Given a forger F winning the UF-KOA game, consider the following reduction P^* breaking IMP-KOA:

1. Upon receiving an instance pk from the IMP-KOA challenger, P^* forwards pk to F.
2. P^* responds to RO queries made by F as the actual RO $\mathcal{O}$ would, except at one query picked uniformly at random: P^* parses the query input as (pk, a', m'), and initiates an impersonation attack using a' as its first message. On receiving challenge e', it programs the RO such that it returns e' on input (pk, a', m').
3. Upon receiving a message-forgery pair $(m^*, (a^*, z^*))$ from F, if it verifies with pk and if (pk, a^*, m^*) was previously forwarded to the IMP-KOA game, P^* outputs z^* as a response. Otherwise, P^* halts with $\perp$.

The above reduction successfully breaks IMP-KOA, albeit with some multiplicative factor of loss proportional to the number of RO queries.

Once we assume IMP-PA security of the underlying ID, it is straightforward to extend the proof to show UF-CMA: whenever the forger makes a sign query with input message m, the reduction consumes one of transcripts (a, e, z) received in the IMP-PA game and programs the RO so that it returns e on input (pk, a, m). This perfectly simulates the forger F's view in the UF-CMA game, and thus one can use F to break IMP-PA in a similar fashion. □

1.3 Overview of Thesis

As reviewed above, public-coin proof systems are a powerful tool in cryptology since they form the foundation of many efficient non-interactive proofs, identifications, digital signatures, etc. The goal of this thesis is to push forward the study of cryptographic constructions based on public-coin protocols in different directions. The main body of this thesis is dissected into two parts: advanced security analyses of existing digital signature schemes, and new constructions supporting extended functionalities. Each part further comprises two independent results, which we summarize below. In all publications the list of authors is sorted alphabetically rather than the extent of individual contributions.

Part I Advanced Security Analysis of Digital Signatures

In this part, we investigate the efficacy of stronger attack models that are *not* covered by the traditional framework of provable security. Notice that the usual security notions for a signature only allow adversaries to have *black-box* access to the signing oracle, meaning that they can neither peek at nor influence internal states of a signer. Although this is a reasonable assumption in theory and often gives rise to elegant security proofs, there is no guarantee that real life attackers respect the aesthetics of our tradition once the schemes are deployed in the wild. So what happens, for example, if the attacker somehow manages

to learn *side-channel information* about private randomness used by the signer in addition to an output signature? Or what if the randomness used by the signer is somewhat *biased*? It is in fact easy to see that the proofs for Theorems 1.1 and 1.2 will miserably fail once a sign query leaks internal states of the Sign function, because the underlying zero knowledge simulator simply cannot provide such information.

It turns out that the consequences are a lot more serious than just "proof not going through"—such *randomness failures* often enable a devastating key recovery attack. To see why, it is instructive to look at the Schnorr scheme [Sch91] as an example. Recall that a prover of the underlying Σ-protocol proceeds as follows: 1) sample uniform $r \in \mathbb{Z}_q$ and output $a = g^r$ as a commit message, 2) receive uniform challenge $e \in \mathbb{Z}_q$, and 3) compute the response $z = r + e \cdot \mathsf{sk} \bmod q$. Suppose the prover reuses the same r across different protocol executions by mistake and thus the same a appears more than once. Since the verifier still freshly samples challenges, the protocol will end up producing transcripts with distinct challenges. The attacker can now abuse special soundness to extract sk by observing two transcripts (a, e, z) and (a, e', z').

The common theme of this part is randomness failures. Randomness failures occur in various ways in practical systems: 1) *side-channel analysis* [Koc96], helping attackers steal secrets like signing keys by exploiting various side-channel leakages from cryptographic devices, such as the time a device spends running sensitive operations, 2) *fault analysis* [BDL97], an active version of side-channel attacks that deliberately causes malfunction in the target device, e.g., via voltage glitching, or 3) *implementation mistakes* involving insecure implementation of random number generator, e.g., the well-known vulnerability of the ECDSA implementation used for signing PlayStation 3 software [Fil11]. As researchers still keep discovering such vulnerabilities in today's deployed systems [MBA+21, BH19, JSSS20, DDE+18, ABuH+19, Rya18, uHGDL+20, MSEH20], it is paramount to further analyze the exact risk of these powerful attacks and propose appropriate countermeasures to prevent them in practice. We summarize contributions of each chapter below.

Chapter 2 *LadderLeak*

This chapter is based on the main body of the following published work [ANT+20a]. The result was also presented at Real World Crypto 2021 and Black Hat Europe 2020. Appendices are deferred to the full version [ANT+20b].

> [ANT+20a] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In *ACM CCS 2020*, pp. 225–242. ACM Press, 2020. DOI: 10.1145/3372297.3417268
>
> [ANT+20b] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. Cryptology ePrint Archive, Report 2020/615, Full version. Available at https://eprint.iacr.org/2020/615.pdf

Background Schnorr and (EC)DSA are discrete log-based signature schemes broadly used in real-world systems. It is well-known that ephemeral randomness of these schemes is extremely sensitive: if the randomness r does not completely follow the uniform distribution over $\mathbb{Z}_q$ or r's partial bits are leaked, an attacker can eventually recover the secret signing key by collecting sufficiently many signatures [Ble00, HGS01]. The key recovery part is

a particular instance of Boneh and Venkatesan's *hidden number problem* (HNP) [BV96]. That observation has been practically exploited in many attacks in the literature, taking advantage of implementation defects or side-channel vulnerabilities in various concrete ECDSA implementations. However, most of the attacks so far have relied on at least *2 bits* of randomness bias.

Contributions This chapter studies the following extreme scenario of the HNP: what if each signing attempt leaks *less than 1 bit* of information about randomness, in the sense that it reveals the most significant bit of one-time randomness, but with probability < 1? We show that even such a mild leakage is exploitable in practice, by giving several theoretical improvements to Bleichenbacher's Fourier analysis approach to solving the HNP [Ble00, Ble05]. We then uncover *LadderLeak*, a novel class of side-channel vulnerabilities in implementations of the Montgomery ladder used in ECDSA scalar multiplication. The vulnerability is in particular present in several versions of OpenSSL. Taken together, we practically break *LadderLeak*-vulnerable ECDSA implementations instantiated over the `sect163r1` and NIST P-192 elliptic curves. In so doing, we achieve several significant computational records in practical attacks against the HNP. Although the original result is tailored to attacks on ECDSA, the improvements to HNP analysis directly affect Schnorr with biased randomness.

Chapter 3 Security of Hedged Fiat-Shamir Signatures

This chapter is based on the main body of the following published work [AOTZ20]. A short remark on HVZK has been inserted before Definition 3.2 and the complete description of OFaultSign is added in Fig. 3.4. Further details are deferred to the full version [AOTZ19].

> [AOTZ20] D. F. Aranha, C. Orlandi, A. Takahashi, and G. Zaverucha. Security of hedged Fiat-Shamir signatures under fault attacks. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 644–674. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45721-1_23
>
> [AOTZ19] D. F. Aranha, C. Orlandi, A. Takahashi, and G. Zaverucha. Security of hedged Fiat-Shamir signatures under fault attacks. Cryptology ePrint Archive, Report 2019/956, Full version. Available at https://eprint.iacr.org/2019/956.pdf

Background To mitigate the catastrophic risk of randomness failure, deterministic generation of one-time randomness was adapted in several signature schemes, including, but not limited to, EdDSA [BDL+12], ECDSA [Por13], and some lattice-based post-quantum signatures [BAA+19, LDK+19]. This solution attempts to avoid biased/reused randomness, by deriving r via a hash function instead of relying on a pseudo random generator (PRG): it computes $r = \mathsf{H}(\mathsf{sk}, m)$, where sk is a signing key and m is a message to be signed.

However, multiple recent studies have practically demonstrated that such de-randomized Fiat-Shamir schemes are vulnerable to differential fault attacks [Bae14, Sch16, BP16, RP17, ABF+18, PSS+18, SB18, BP18, RJH+19]. The attack is simple, yet quite devastating: essentially, a fault attacker can perform a full key recovery after observing one legitimate signature σ on m, and one faulty signature $\tilde{\sigma}$ on the same m, which shares the randomness with σ due to the deterministic nature. In order to balance concerns of both randomness failures and the threat of fault injection, some signature designs [Per16, ZCD+19, qTE19] are advocating a "hedged" derivation of randomness: they compute $r = \mathsf{H}(\mathsf{sk}, m, n)$, where n is an additional *nonce*. The benefit of randomness hedging is that the fault attack can

be prevented as long as a nonce n is non-repeating, while n does not need to be completely uniform. Despite the growing popularity of the hedged paradigm in practical signature schemes, to the best of our knowledge, there has been no attempt to formally analyze the fault resilience of hedged signatures.

Contribution This chapter involves detailed security analyses of the fault resilience of hedged signatures constructed from three-round public-coin protocols in the random oracle model, using the methodology of provable security. We first expand the usual UF-CMA security game by incorporating an adversarial model characterizing bit-tampering fault attacks, and investigate their impact across different steps of the signing operation. Under the proposed model we formally prove that for some types of faults, attacks are mitigated by the hedged paradigm, while attacks remain possible for others. As concrete case studies, we then apply our results to XEdDSA [Per16], a hedged version of EdDSA used in the Signal messaging protocol, and to Picnic2 [ZCD+19], a hedged Fiat-Shamir signature scheme in Round 2 of the NIST Post-Quantum standardization process.

Part II New Constructions Supporting Advanced Functionalities

In this part, we propose new constructions stemming from public-coin protocols. Although non-interactive proofs, identifications, and signatures are the applications most typically appearing in real world systems, the power of public-coin protocols is not limited to these basic primitives. For example, what if you would like to *distribute* the task of signing to multiple computers in such a way that it is infeasible to produce a legitimate signature without everyone's contribution? A so-called *multi-party distributed signature* offers a solution to this scenario. This advanced functionality should introduce some useful buffer in case of key theft. Clearly, the attacker against usual signatures can completely compromise security by corrupting one single signer, whereas this is not the case anymore against multi-party signatures thanks to the ability to distribute trust. Multi-party signatures have been studied for a long time in the literature, taking several different forms depending on the purpose and security model, e.g., *threshold signatures*, *multi-signatures*, etc. Perhaps motivated by new use cases, such as distributed wallets for cryptocurrencies, the study of Fiat-Shamir-based interactive multi-party signatures is an active research area in recent years [MPSW19, DEF+19, NRS21, AB21, BD21, KG20, GKMN21, KMOS21]. However, the majority of these schemes are Schnorr-based and thereby do not withstand the celebrated polynomial-time algorithm of Shor [Sho94].

Public-coin proofs are also known to be useful for adding *verifiability* to encryption schemes. Say you as a cloud service administrator want to audit certain encryption operations performed by your cloud systems, e.g., think of the situation where some secret signing key must be securely exported from one cloud instance to another via public-key encryption, while the secret itself is not to be exposed outside the trust boundary of cloud services. This can be naïvely achieved via general-purpose proof systems: the sender just produces a ZKPoK of the secret w (i.e., a witness), and the administrator verifies a proof to be convinced that a ciphertext c is produced via correct computation of $\mathsf{Enc}(w)$ and that the preimage w indeed corresponds to the public signature verification key x (i.e., a statement). However, whether one can efficiently instantiate such ZKPoK highly depends on particulars of the encryption function, and straightforward application of general-purpose ZKPoK tends to get expensive if the function $\mathsf{Enc}()$ internally performs mixed arithmetic operations (e.g., hashing) and algebraic computations (e.g., scalar multiplication over an elliptic curve).

This is why several previous works presented *verifiable encryption* [Sta96, CD00, CS03, LN17, LCKO19], a tailor-made primitive more effectively combining specific encryption schemes and proof systems, rather than relying on a monolithic ZKPoK to prove everything in one go. The motivation is somewhat analogous to *commit-and-prove* frameworks of [CGM16, AGM18, BHH+19, CFQ19, CFF+21, ABC+22] that combine different ZK proof systems depending on the nuances of a computation, but not many generic methods are known in context of verifiable encryption.

Motivated by these scenarios, the following chapters propose novel constructions supporting advanced functionalities.

Chapter 4 Two-Round Multi-Party Signing from Lattices

This chapter is based on [DOTT22], which is a journal version of [DOTT21] and one of the invited papers from PKC 2021. To avoid redundancy, appendices and security proof for the two-round multi-signature scheme are deferred to the full version [DOTT20].

> [DOTT21] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC 2021, Part I*, vol. 12710 of *LNCS*, pp. 99–130. Springer, Heidelberg, 2021. DOI: 10.1007/978-3-030-75245-3_5
>
> [DOTT22] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *J. Cryptol.*, 35(14), 2022. DOI: 10.1007/s00145-022-09425-3
>
> [DOTT20] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. Cryptology ePrint Archive, Report 2020/1110, Full version. Available at https://eprint.iacr.org/2020/1110.pdf

Background Although fairly efficient, most recent multi-party Fiat-Shamir signatures are based on Schnorr and thereby are not post-quantum secure. At a high-level all these schemes essentially exploit homomorphism of the Schnorr proofs: 1) each party i owning $\mathsf{sk}_i \in \mathbb{Z}_q$ announces the first "commit" message $a_i = g^{r_i}$, 2) computes the product $a = \prod_i a_i$ on receiving other parties' commit messages, 3) derives challenge $e = \mathsf{H}(\mathsf{pk}, a, m)$, 4) computes the response $z_i = r_i + e \cdot \mathsf{sk}_i \bmod q$, and 5) outputs a and $z = \sum_i z_i \bmod q$ as a signature. Clearly, such a signature can be verified with a combined public key $\mathsf{pk} = \prod_i g^{\mathsf{sk}_i}$ using the usual Schnorr verification algorithm.

A natural way to adapt this idea in the post-quantum setting would be to extend *the Fiat–Shamir with Aborts (FSwA)* lattice-based proofs and signatures due to Lyubashevsky [Lyu09, Lyu12]. The paradigm shares its basic structure with Schnorr's Σ-protocol, with the important difference that the prover may abort the protocol after seeing the challenge, resulting in a significant reduction of proof/signature size. Several previous works in fact proposed FSwA distributed multi-party signing following Schnorr-based constructions. However, these protocols involve at least three rounds of interaction among signers as opposed to two as in the above basic Schnorr-based scheme. Moreover, due to subtle technical issues caused by aborts they either require non-standard hardness assumptions or lack complete security proofs.

Contribution The chapter comprises new multi-party, distributed signing protocols in the random oracle model relying on standard lattice-based hardness assumptions. We study two similar classes of distributed signing, namely, n-out-of-n signature and multi-signature, both of which allow a group of signers to jointly produce a single signature on the same message. We realize relatively simple lattice-based protocols by carefully lifting several existing tricks known in the discrete log setting into the lattice world. Our approach makes use of lattice-based homomorphic commitments both to reach near-optimal round complexity and to fully circumvent the technical issues inherent in security proof for FSwA multi-party signatures. This allows us to describe the first *two-round* protocols whose security can be formally reduced to the hardness of short integer solution (SIS) and learning with errors (LWE) problems.

Chapter 5 Verifiable Encryption from MPC-in-the-Head

This chapter is based on the main body of a manuscript currently in submission. Section 5.1.2 has been replaced with a more exhaustive survey of related work and appendices are deferred to the following full version.

> [TZ21] A. Takahashi and G. Zaverucha. Verifiable encryption from MPC-in-the-Head. Cryptology ePrint Archive, Report 2021/1704, https://eprint.iacr.org/2021/1704.pdf

Background Verifiable encryption (VE) is a protocol where one can provide assurance that an encrypted plaintext satisfies certain properties or relations. Similar to proof systems VE protocols involve a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, and in addition a *receiver* $\mathcal{R}$. A VE receiver $\mathcal{R}$ generates a key pair $(\mathsf{pk}, \mathsf{sk})$ for some public key encryption scheme and publishes pk. A prover $\mathcal{P}$ encrypts a plaintext (or witness) w using pk to ciphertext c and interacts with $\mathcal{V}$ to convince him/her that c correctly decrypts to a right witness corresponding to some public statement x. Security requirements for VE roughly say that $\mathcal{R}$ be able to recover a right witness as long as $\mathcal{V}$ outputs 1, whereas $\mathcal{V}$ without the knowledge of decryption key learns nothing about the encrypted plaintext. It is an important building block in cryptography with many useful applications, such as key escrow, group signatures, optimistic fair exchange, and others.

Although several VE constructions exist in the literature, the majority of them are either restricted to instantiation with specific public-key encryption schemes or relations, for which efficient "zero knowledge proof of plaintext knowledge" exist. On the other hand, the only *generic construction* of VE supporting any public-key encryption schemes dates back to Camenisch and Damgård [CD00]. Roughly, their protocol proceeds as follows: $\mathcal{P}(\mathsf{pk}, x, w)$ and $\mathcal{V}(\mathsf{pk}, x)$ execute a Σ-protocol with $\{0,1\}$-challenge for particular relation of interest, except that $\mathcal{P}$ now additionally sends $c_e = \mathsf{Enc}(\mathsf{pk}, z_e)$ in the first flow for both $e \in \{0,1\}$ and opens one of the responses with the corresponding encryption randomness during the third flow. The receiver $\mathcal{R}$ then gets the entire transcript, decrypts the remaining ciphertext, and recovers a witness by exploiting special soundness of the Σ-protocol. However, their approach may not be satisfactory for some scenarios in that 1) it is limited to Σ-protocols with 1-bit challenge space, which do not necessarily cover modern, efficient proof systems, and 2) the analysis lacks effects of decryption failure, which is becoming relevant due to many post-quantum encryption schemes with imperfect correctness, such as ones based on lattices.

Contributions In this work, we propose a novel framework that realizes VE protocols using zero-knowledge proof systems based on the MPC-in-the-head paradigm [IKOS07]. We describe a generic compiler turning a large class of MPC-in-the-head proofs into secure VE protocols for any secure public-key encryption scheme, including ones with imperfect correctness. As in [CD00], our approach completely circumvents proof of plaintext knowledge and thus the work of the prover can be focused on proving the encrypted data satisfies the relation. We further show that our compiler is compatible with several efficient MPC-in-the-head proofs that do not fall into the category of Σ-protocols. Our VE constructions can be made non-interactive in the random oracle model via the usual Fiat-Shamir transform. As concrete applications we describe new approaches to verifiably encrypting discrete logarithms in any prime order group and AES private keys.

1.4 Other Publications

The author has contributed to the following results which are not included in this thesis. [TTA18a], [TT19], [ABE+21], and [GOP+22] are in the domain of advanced security analysis of existing schemes, whereas [ABC+22], [EFG+22], and [BTT22] propose new constructions.

1.4.1 Space Optimization of Bleichenbacher's Attack

[TTA18a] A. Takahashi, M. Tibouchi, and M. Abe. New Bleichenbacher Records: Fault Attacks on qDSA Signatures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):331–371, 2018. DOI: 10.13154/tches.v2018.i3.331-371, Full version available at https://eprint.iacr.org/2018/396.pdf

1.4.2 Fault Attacks on Elliptic Curve Cryptography

[TT19] A. Takahashi and M. Tibouchi. Degenerate fault attacks on elliptic curve parameters in OpenSSL. In *IEEE EuroS&P 2019*, pp. 371–386. IEEE, 2019. DOI: 10.1109/EuroSP.2019.00035, Full version available at https://eprint.iacr.org/2019/400.pdf

1.4.3 Side-channel Attacks on MPC-in-the-Head Signatures and Countermeasures

[ABE+21] D. F. Aranha, S. Berndt, T. Eisenbarth, O. Seker, A. Takahashi, L. Wilke, and G. Zaverucha. Side-channel protections for Picnic signatures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):239–282, 2021. DOI: 10.46586/tches.v2021.i4.239-282, Full version available at https://eprint.iacr.org/2021/735.pdf. Preliminary version appeared at the 3rd NIST PQC Standardization Conference

1.4.4 Compiler for Commit-and-Prove SNARKs

[ABC+22] D. F. Aranha, E. M. Bennedsen, M. Campanelli, C. Ganesh, C. Orlandi, and A. Takahashi. ECLIPSE: Enhanced Compiling Method for Pedersen-Committed zkSNARK Engines. In *PKC 2022*, vol. 13177 of *LNCS*, pp. 584–614. Springer, 2022. DOI: 10.1007/978-3-030-97121-2_21, Full version available at https://eprint.iacr.org/2021/934.pdf

1.4.5 Non-malleability of Fiat-Shamir Proof Systems

[GOP+22] C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model). In *EUROCRYPT 2022*, vol. 13276 of *LNCS*, pp. 397–426. Springer, 2022. DOI: 10.1007/978-3-031-07085-3_14, Full version available at https://eprint.iacr.org/2021/1393.pdf

1.4.6 Better Hash-and-Sign Signatures from Lattices

[EFG+22] T. Espitau, P. Fouque, F. Gérard, M. Rossi, A. Takahashi, M. Tibouchi, A. Wallet, and Y. Yu. Mitaka: A simpler, parallelizable, maskable variant of Falcon. In *EUROCRYPT 2022*, vol. 13277 of *LNCS*, pp. 222–253. Springer, 2022. DOI: 10.1007/978-3-031-07082-2_9, Full version available at https://eprint.iacr.org/2021/1486.pdf. Preliminary version appeared at the 3rd NIST PQC Standardization Conference

1.4.7 Offline-Online Multi-Signature from Lattices

[BTT22] C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *CRYPTO 2022*, LNCS. Springer, To appear

Part I

Advanced Security Analysis

Chapter 2

LadderLeak

2.1 Introduction

The ECDSA algorithm is one of the most widely deployed signature schemes today, and is part of many practical cryptographic protocols such as TLS and SSH. Its signing operation relies on an ephemeral random value called *nonce*, which is particularly sensitive: it is crucial to make sure that the nonces are kept in secret *and* sampled from the uniform distribution over a certain integer interval. It is easy to see that if the nonce is exposed or reused completely, then an attacker is able to extract the secret signing key by observing only a few signatures. By extending this simple observation, cryptanalysts have discovered stronger attacks that make it possible to recover the secret key even if short bit substrings of the nonces are leaked or biased. These extended attacks relate key recovery to the so-called hidden number problem (HNP) of Boneh and Venkatesan [BV96], and are part of a line of research initiated by Howgrave-Graham and Smart [HGS01], who described a lattice-based attack to solve the corresponding problem, and Bleichenbacher [Ble00], who proposed a Fourier analysis-based approach.

Lattice-based attacks are known to perform very efficiently when sufficiently long substrings of the nonces are known to the attacker (say over 4 bits for signatures on a 256-bit elliptic curve, and at least 2 bits for a 160-bit curve). As a result, a number of previous works adapted Howgrave-Graham and Smart's technique to practically break vulnerable implementations of ECDSA and related schemes like Schnorr signatures [Sch90], for instance by combining it with side-channel analysis on the nonces (see related works in Section 2.2.4). However, a limitation of lattice-based attacks is that they become essentially inapplicable when only a very small fraction of the nonce is known for each input sample. In particular, for a single-bit nonce leakage, it is believed that they should fail with high probability, since the lattice vector corresponding to the secret is no longer expected to be significantly shorter than other vectors in the lattice [NT12, AFG+14]. In addition, lattice-based approaches assume that inputs are perfectly correct, and behave very poorly in the presence of erroneous leakage information.

In contrast, Bleichenbacher's Fourier analysis-based attack can in principle tackle arbitrarily small nonce biases, and handles erroneous inputs out of the box, so to speak. Despite those features, it has garnered far less attention from the community than lattice-based approaches, perhaps in part due to the lack of formal publications describing the attack until recently (even though some attack records were announced publicly [Ble05]). It was only in 2013 that De Mulder et al. [DHMP13] revisited the theory of Bleichenbacher's

Table 2.1: Comparison with the previous records of solutions to the hidden number problem with small nonce leakages. Each row corresponds to the size of group order in which the problem is instantiated. Each column corresponds to the *maximum* number of leaked nonce bits per signature. Citations in green (resp. purple) use Bleichenbacher's approach (resp. lattice attacks).

	< 1	1	2	3	4
256-bit	–	–	[TTA18a]	[TTA18a]	[Rya18, Rya19, MSEH20, WSBS20]
192-bit	This work	This work	–	–	–
160-bit	This work	[AFG+14, Ble05], this work (less data)	[Ble00], [LN13]	[NS02]	–

approach in a formally published scholarly publication, followed soon after by Aranha et al. [AFG+14], who overcame the 1-bit "lattice barrier" by breaking 160-bit ECDSA with a single bit of nonce bias. Takahashi, Tibouchi and Abe [TTA18a] improved the space complexity of Bleichenbacher's attack and broke qDSA [RS17] (a variant of Schnorr) with 2-bit nonce leaks. However, the practicality of these works may seem limited: indeed, De Mulder et al. attacked parameters (i.e. 384-bit with 5-bit bias) that can be solved more efficiently by lattice attacks; Aranha et al. required over 8 billion signatures as input; and Takahashi et al. only mounted the attack by artificially injecting physical faults into the modified, non-standard implementation.

In this work, we present the first real-world ECDSA vulnerabilities that are *not* susceptible to lattice attacks, but become practically exploitable with the Fourier analysis method, thanks to novel theoretical improvements that we propose over existing literature.

2.1.1 Contributions

New Cache Timing Attacks Against OpenSSL Montgomery Ladder We first propose *LadderLeak*, a new class of vulnerabilities lurking in scalar multiplication algorithms invoked by ECDSA. Our attack exploits small timing differences within the implementations of Montgomery ladder [Mon87] relying on inappropriate coordinate handling, and allows the attacker to learn a single-bit of the secret scalar (which corresponds to nonces in ECDSA) with high probability. We discovered *LadderLeak* vulnerabilities in several versions of OpenSSL (particularly in the `1.0.2` and `1.1.0` branches), and in version `0.4.0` of RELIC toolkit [A+]. We present two attack flavors: one for binary curves and the other for prime curves. Both have been experimentally validated, and provide high-precision distinguishers for ECDSA nonces in our target versions of OpenSSL. In principle, the vulnerability affects various curve parameters in the above implementations, including NIST P-192, P-224, P-256, P-384, P-521, B-283, K-283, K-409, B-571, `sect163r1`, `secp192k1`, `secp256k1` [1]. In Section 2.3 we describe the attack idea, as well as concrete side-channel experiments carried out using Flush+Reload cache timing attacks [YF14]. As concrete targets we choose ECDSA instantiated over NIST P-192 and `sect163r1`, and successfully retrieve 1-bit of nonce information with high probability.

[1] OpenSSL does not invoke the vulnerable ladder implementation for P-256 by default and relies instead on custom code enabled during compilation. Custom code can also be similarly enabled at build time for P-224 and P-521 with the switch `enable-ec_nistp_64_gcc_128` (see `https://wiki.openssl.org/index.php/Compilation_and_Installation#Configure_Options`). We thank an anonymous reviewer for pointing out these behaviors of OpenSSL.

Improved Theoretical Analysis of Bleichenbacher's Solution to the HNP In Section 2.4 we establish a unified time–space–data tradeoff formula for Bleichenbacher style attacks, and use it to concretely find optimal attack parameter choices for a given group size and a given amount of nonce bias. Our formula relies on the connection between the hidden number problem on the one hand and the $\mathcal{K}$-list sum problem on the other (the latter of which is a sub-problem of Wagner's generalized birthday problem (GBP) [Wag02], particularly well-known in symmetric key cryptology). Our approach is generic, allowing to integrate in principle any $\mathcal{K}$-list integer sum algorithms to derive a similar tradeoff formula. Although Bleichenbacher's method was thought to require billions of signatures as input to attack 1-bit of nonce leakage, we prove that it is possible to significantly reduce the data complexity by carefully choosing the inputs to our tradeoff formula. Our analysis also provides significant improvements to the data complexity for leaks of more than 1 bit, allowing the side-channel attacker to recover the ECDSA key given only several thousands signatures in many cases, or even several hundreds in some scenarios. The complete complexity estimates given in [ANT+20b] may therefore be of independent interest. We further incorporate the effect of misdetection in the most significant bit of the HNP samples, which becomes crucial when combining with the practical side-channel leakage we consider.

Optimized Implementation and New Attack Records for the HNP Putting both contributions together, we mount a full signing key recovery attack on ECDSA signatures instantiated over the `sect163r1` binary curve and over the NIST P-192 prime curve, using *less* than 1 bit of nonce leakage (in the sense that we recover 1 bit of the nonces, but with probability less than 1). The tradeoff formula we develop allows us to break the corresponding HNP instances with realistic computational resources. In our attack experiments, presented in Section 2.5, the data complexity required for the former case is significantly less than Aranha et al. [AFG+14], by a factor of around 2^{10}. Furthermore, to the best of our knowledge, 192-bit ECDSA has never been broken before with 1 bit of leakage or less (see Table 2.1 for the comparison with previous HNP records), and our concrete attack therefore marks a dramatic advance in concrete attacks on the HNP. This was made possible by tuning the tradeoffs to optimize the time complexity and by running our highly optimized scalable implementation in Amazon Web Service (AWS) EC2 cloud instances. The approach and implementation devised in this work can be applied to various types of leakage from ECDSA independent of the *LadderLeak* vulnerability, and hence offer an interesting avenue for future cryptanalytic work. For example, our empirical results also indicate that breaking larger instances like P-224 with 1-bit leak, or P-256 with 2-bit leak would be practically doable with relatively modest data complexity. The experimental results regarding cache attacks, submitted patches with countermeasures, tradeoff formula solver, and optimized implementation of Bleichenbacher's attack are available in our GitHub repository[2].

2.1.2 Vulnerable Software Versions and Coordinated Disclosure

In December 2019, we originally reported to the OpenSSL development team the vulnerabilities in versions `1.0.2t` and `1.0.1l` in accordance with the OpenSSL security policy[3] before the end of long-term support for those versions. After version `1.0.2u` was released, we confirmed that the same vulnerabilities were still present, and hence we proposed a

[2] https://github.com/akiratk0355/ladderleak-attack-ecdsa
[3] https://www.openssl.org/policies/secpolicy.html

patch with corresponding countermeasures, which has already been approved[4]. Although the `1.0.2` branch is now out of public support as of May 2020, the OpenSSL development team still provides its customers with an extended support for `1.0.2` and our patch is included in their further internal releases. While it is hard to estimate how the vulnerability would affect real products due to the lack of public releases containing the patch, searching over GitHub revealed several projects updating their build process to take into account a new release from the `1.0.2` branch. Similar steps have also been taken to address the vulnerability in RELIC, and a fix was pushed in January 2020.

2.2 Preliminaries

Notations We denote the imaginary unit by roman i. For any positive integer x a function $\mathtt{MSB}_n(x)$ returns its most significant n bits. When a and b are integers such that $a < b$ we use the integer interval notation $[a, b]$ to indicate $\{a, a+1, \ldots, b\}$. Throughout the paper $\log c$ denotes the binary logarithm of $c > 0$.

2.2.1 Cache Attacks

To deliver the high performance expected of modern computers, processors employ an array of techniques that aim to predict program behavior and optimize the processor for such behavior. As a direct consequence, program execution affects the internal state of the processor, which in turn affects the speed of future program execution. Thus, by monitoring its own performance, a program can learn about execution patterns of other programs, creating an unintended and unmonitored communication channel [GYCH18].

Unintended leakage of cryptographic software execution patterns can have a devastating impact on the security of the implementation. Over the years, multiple attacks have demonstrated complete key recovery, exploiting leakage through the internal state of various microarchitectural components, including caches [TTMH02, OST06, ABG10, LYG+15], branch predictors [AGS07, LSG+17], functional units [ABuH+19, AS07], and translation tables [GRBG18].

Flush+Reload [GBK11, YF14] is a prominent attack strategy, in which the attacker monitors victim access to a memory location. The attack consists of two steps. In the *flush* step, the attacker evicts the monitored memory location from memory, typically using a dedicated instruction such as CLFLUSH. The attacker then waits a bit to allow the victim time to execute. Finally, in the *reload* step, the attacker accesses the memory location, while measuring how long the access takes. If the victim has not accessed the monitored location, it will remain uncached, and access will be slow. Conversely, if the victim has accessed the monitored location between the flush and the reload steps, the memory location will be cached, and access will be fast. Repeating the attack, an attacker can recover the memory usage patterns of the victim, allowing attacks on symmetric ciphers [GBK11, IAIES14], public key primitives, both traditional [YF14, GB17, BBG+17] and post-quantum [BHLY16, PBY17], as well as on non-cryptographic software [GSM15, YFT20].

[4] https://github.com/openssl/openssl/pull/11361

2.2.2 The Montgomery Ladder and its Secure Implementation

An elliptic curve E defined over a finite field $\mathbb{F}$ is the set of solutions $(x, y) \in \mathbb{F}$ that satisfy the curve equation, together with a point at infinity $\mathcal{O}$. The chord-and-tangent rule defines a group law ($\oplus$) for adding and doubling points on the curve, with $\mathcal{O}$ the identity element. Given $P \in E(\mathbb{F})$ and $k \in \mathbb{Z}$, the *scalar multiplication* operation computes $R = [k]P$, which corresponds to adding P to itself $(k-1)$ times. Cryptographic protocols rely on multiplication by a secret scalar as a fundamental operation to base security on the Elliptic Curve Discrete Logarithm Problem (ECDLP): find k given inputs $(P, [k]P)$. Concrete instances of elliptic curves used in cryptography employ a subgroup of points of large prime order q for which the ECDLP is known to be hard. For efficiency reasons, it is common to represent points in projective coordinates (X, Y, Z) and avoid the computation of expensive field inversions during the evaluation of the group law.

In many settings, an adversary able to recover leakage from the scalar multiplication operation, in particular bits of the scalar, can substantially reduce the effort necessary to solve the ECDLP. The Montgomery ladder, which was initially proposed for accelerating the ECM factorization method [Mon87], later became crucial for secure implementations of elliptic curve cryptography due to its inherent regularity in the way the scalar is processed: the same number of group operations is required no matter the bit pattern of k. Algorithm 1 illustrates the idea. There is a rich literature about coordinate systems and elliptic curve models for securely implementing the algorithm [CS18, OLR18]. Unfortunately, these techniques do not always work for the standardized curves in the Weierstrass model that greatly contributed to the adoption of elliptic curves in industry through the SECG and NIST standards [KKM08].

A *constant-time* implementation of the Montgomery ladder protected against timing attacks must satisfy three basic preconditions: (i) the number of loop iterations must be fixed; (ii) memory operations cannot depend on bits of the secret scalar, to avoid leakage through the memory hierarchy; (iii) the group law must be evaluated with the same number and type of field operations in the same order, independently of the bits of the scalar. The first is easier to guarantee, by conditionally adding q or $2q$ until the resulting scalar $\hat{k}$ has fixed length. Note that this approach has the interesting side-effect of preserving the MSB of k in the second MSB of $\hat{k}$ when q is just below a power of 2. The second can be achieved by simply replacing branches with conditional operations to swap the accumulators (R_0, R_1). The third is more involved, but greatly simplified by *complete* addition laws that compute the correct result for all possible inputs [BL07] (even when the point is being doubled) without any exceptions or corner cases. While complete addition laws for Weierstrass curves do exist [RCB16], they incur a substantial performance penalty and have not been popularized enough for current implementations of classical standardized curves. A version of the algorithm with these countermeasures applied can be found in [ANT+20b]. As we further discuss in Section 2.3, attempts to improve side-channel security of current implementations of scalar multiplication risk retrofitting an otherwise constant-time Montgomery ladder on top of a group law implementation that still leaks information through optimizations.

2.2.3 ECDSA and Hidden Number Problem with Erroneous Input

The signing key extraction from the nonce leakages in ECDSA signatures typically amounts to solving the so-called *hidden number problem (HNP)*. We present a generalized variant

Algorithm 1 Montgomery ladder

Require: $k \in \mathbb{Z}_q$, point P on $E(\mathbb{F})$.
Output: The results of the scalar multiplication $R = [k]P$.

$(R_0, R_1) \leftarrow (P, 2P)$
for $i = \lfloor \lg(k) \rfloor - 1$ **downto** 0 **do**
  if $k_i = 0$ **then**
    $(R_0, R_1) \leftarrow ([2]R_0, R_0 \oplus R_1)$
  else
    $(R_0, R_1) \leftarrow (R_0 \oplus R_1, [2]R_1)$
  end if
end for
return R_0

Algorithm 2 ECDSA signature generation

Require: Signing key $\mathsf{sk} \in \mathbb{Z}_q$, message $\mathtt{msg} \in \{0,1\}^*$, group order q, base point G, and cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
Output: A valid signature (r, s)

$k \xleftarrow{\$} \mathbb{Z}_q$
$R = (r_x, r_y) \leftarrow [k]G; r \leftarrow r_x \mod q$
$s \leftarrow (H(\mathtt{msg}) + r \cdot \mathsf{sk})/k \mod q$
return (r, s)

of the original HNP by Boneh and Venkatesan [BV96], incorporating some erroneous information of the most significant bits. In particular, the error distribution below models the attacker's misdetection during the side-channel acquisition.

Definition 2.1 (Hidden Number Problem with Erroneous Input)**.** *Let q be a prime and $\mathsf{sk} \in \mathbb{Z}_q$ be a secret. Let h_i and k_i be uniformly random elements in $\mathbb{Z}_q$ for each $i = 1, \ldots, M$ and define $z_i = k_i - h_i \cdot \mathsf{sk} \mod q$. Suppose some fixed distribution χ_b on $\{0,1\}^b$ for $b > 0$ and define a probabilistic algorithm $\mathtt{EMSB}_{\chi_b}(x)$ which returns $\mathtt{MSB}_b(x) \oplus e$ for some error bit string e sampled from χ_b. Given (h_i, z_i) and $\mathtt{EMSB}_{\chi_b}(k_i)$ for $i = i, \ldots, M$,* the HNP with error distribution χ_b *asks one to find* sk.

In our concrete attacks against OpenSSL ECDSA, we focus on the case where $b = 1$ and χ_b is the Bernoulli distribution $\mathscr{B}_\epsilon$ for some error rate parameter $\epsilon \in [0, 1/2]$, i.e. $\mathtt{EMSB}_{\chi_b}(x)$ simply returns the negation of the most significant bit of x with probability ϵ, and otherwise returns the correct bit.

A straightforward calculation shows that a set of ECDSA signatures with leaky nonces is indeed an instance of the HNP. Notice that the ECDSA signature (r, s) generated as in Algorithm 2 satisfies $s = (H(\mathtt{msg}) + r \cdot \mathsf{sk})/k \mod q$ for uniformly chosen $k \in \mathbb{Z}_q$. Rearranging the terms, we get

$$H(\mathtt{msg})/s = k - (r/s) \cdot \mathsf{sk} \mod q.$$

Hence letting $z = H(\mathtt{msg})/s \mod q$ and $h = r/s \mod q$, we obtain a HNP sample if the MSB of k is leaked with some probability.

2.2.4 Lattice Attacks on HNP

Boneh and Venkatesan [BV96] suggest solving the Hidden Number Problem by first reducing it to the lattice Closest Vector Problem (CVP). Howgrave-Graham and Smart [HGS01] show the reduction of partial nonce leakage from DSA to HNP, which they solve by reduction to CVP. Nguyen and Shparlinski [NS02] prove that the Howgrave-Graham and Smart approach works with a leak of $\log \log q$ bits from a polynomial number of ephemeral keys. They later extend the result to ECDSA [NS03].

Brumley and Tuveri [BT11] demonstrate a timing attack on OpenSSH acquiring a small number of bits from the ephemeral keys. Following works present electromagnetic emanation [GPP+16, BFMT16], cache [BvSY14, vSY15, ABF+16, GB17] and other microarchitectural [ABuH+19] attacks, all use a conversion of HNP to a lattice problem. Dall et al. [DDE+18] explore the viability of solving HNP with errors using a lattice attack. One of the main target curves in our work is NIST P-192, which was also exploited by Medwed and Oswald [MO09] using several bits of nonce acquired via template-based SPA attacks.

2.2.5 Bleichenbacher's Attack Framework

The Fourier analysis-based approach to the HNP was first proposed by Bleichenbacher [Ble00] and it has been used to break ECDSA, Schnorr and variants with small nonce leakages that are hard to exploit with the lattice-based method [AFG+14, DHMP14, TTA18a]. This section covers the fundamentals of Bleichenbacher's framework, summarized in Algorithm 3. For more theoretical details we refer to the aforementioned previous works. The essential idea of the method is to quantify the modular bias of nonce k using the *bias functions* in the form of inverse discrete Fourier transform (iDFT).

Definition 2.2 (Bias Functions)**.** *Let* $\mathbf{K}$ *be a random variable over* $\mathbb{Z}_q$*. The* modular bias $B_q(\mathbf{K})$ *is defined as*

$$B_q(\mathbf{K}) = \mathrm{E}[e^{(2\pi \mathbf{K}/q)i}]$$

where $E(\mathbf{K})$ *represents the mean and* i *is the imaginary unit. Likewise, the* sampled bias *of a set of points* $K = \{k_i\}_{i=1}^{M}$ *in* $\mathbb{Z}_q$ *is defined by*

$$B_q(K) = \frac{1}{M} \sum_{i=1}^{M} e^{(2\pi k_i/q)i}.$$

When the l MSBs of $\mathbf{K}$ are fixed to some constant and $\mathbf{K}$ is otherwise uniform modulo q, then it is known that the norm of bias $|B_q(\mathbf{K})|$ converges to $2^l \cdot \sin(\pi/2^l)/\pi$ for large q [TTA18a, Corollary 1]. The estimate holds for the sampled bias $B_q(K)$ as well for a given set of biased nonces $\{k_i\}_{i=1}^{M}$. For example, if the first MSB of each k_i is fixed to a constant bit then the bias is estimated as $|B_q(K)| \approx 2/\pi \approx 0.637$. Moreover, if the k_i's follow the uniform distribution over $\mathbb{Z}_q$ then the mean of the norm of sampled bias is estimated as $1/\sqrt{M}$, which is a direct consequence of the well-known fact about average distance from the origin for a random walk on the complex plane [AFG+14].

Small linear combinations Given such a function, it would be straightforward to come up with a naive approach to find sk: for each candidate secret key $w \in \mathbb{Z}_q$, compute the corresponding set of candidate nonces $K_w = \{z_i + h_i w \mod q\}_{i=1}^{M}$ and then conclude that

Algorithm 3 Bleichenbacher's attack framework

Require:
$\{(h_i, z_i)\}_{i=1}^{M}$ - HNP samples over $\mathbb{Z}_q$.
M' - Number of linear combinations to be found.
L_{FFT} - FFT table size.
Output: Most significant bits of sk

Collision search
Generate M' samples $\{(h'_j, z'_j)\}_{j=1}^{M'}$, where $(h'_j, z'_j) = (\sum_i \omega_{i,j} h_i, \sum_i \omega_{i,j} z_i)$ is a pair of linear combinations with the coefficients $\omega_{i,j} \in \{-1, 0, 1\}$, such that for $j \in [1, M']$
1. *Small*: $0 \le h'_j < L_{\mathrm{FFT}}$ and
2. *Sparse*: $|B_q(\mathbf{K})|^{\Omega_j} \gg 1/\sqrt{M'}$ for all $j \in [1, M']$, where $\Omega_j \coloneqq \sum_i |\omega_{i,j}|$.

Bias Computation
$Z \coloneqq (Z_0, \ldots, Z_{L_{\mathrm{FFT}}-1}) \leftarrow (0, \ldots, 0)$
for $j = 1$ to M' **do**
    $Z_{h'_j} \leftarrow Z_{h'_j} + e^{(2\pi z'_j/q)\mathrm{i}}$
end for
$\{B_q(K_{w_i})\}_{i=0}^{L_{\mathrm{FFT}}-1} \leftarrow \mathrm{FFT}(Z)$, where $w_i = iq/L_{\mathrm{FFT}}$.
Find the value i such that $|B_q(K_{w_i})|$ is maximal.
Output most significant $\log L_{\mathrm{FFT}}$ bits of w_i.

$w = \mathsf{sk}$ if the sampled bias $|B_q(K_w)|$ shows a peak value. This is of course no better than the exhaustive search over the entire $\mathbb{Z}_q$. To avoid this issue, the so-called *collision search* of input samples is required, which is the crucial preliminary step to expand the peak width. De Mulder et al. [DHMP14] and Aranha et al. [AFG+14] showed that the peak width broadens to approximately q/L_{FFT}, by taking linear combinations of input samples $\{(h_i, z_i)\}_{i=1}^{M}$ to generate new samples $\{(h'_j, z'_j)\}_{j=1}^{M'}$ such that $h'_j < L_{\mathrm{FFT}}$. This way, one could hit somewhere in the broadened peak by only checking the sampled biases at L_{FFT} candidate points over $\mathbb{Z}_q$. As the inverse DFT at L_{FFT} points can be efficiently computed by fast Fourier transform (FFT) algorithms in $O(L_{\mathrm{FFT}} \log L_{\mathrm{FFT}})$ time and $O(L_{\mathrm{FFT}})$ space, the first goal of the collision search phase is to find sufficiently small linear combinations of the samples so that the FFT on the table of size L_{FFT} becomes practically computable. A few different approaches have been explored to the collision search phase, such as lattice reduction [DHMP14], sort-and-difference [AFG+14] and 4-list sum algorithm [TTA18a].

Sparse linear combinations One may be tempted to repeat such collision search operations as many times as needed until the linear combinations with the desired upper bound are observed. However, the linear combinations come at a price; in exchange of the broader peak width, the peak *height* gets reduced exponentially. Concretely, if the original modular bias of nonce is $|B_q(\mathbf{K})|$ and all coefficients in the linear combinations are restricted to $\{-1, 0, 1\}$ then the peak bias gets exponentiated by L_1-norm of the coefficient vector [TTA18a]. Thus, for the peak height to be distinguishable from the noise value the diminished peak should be significantly larger than the noise value (which is $1/\sqrt{M'}$ on average as mentioned above). This imposes another constraint on the collision search phase: the *sparsity* of linear combinations. In summary, the efficiency of Bleichenbacher's attack crucially relies upon the complexities of small and sparse linear combination search algorithm.

Algorithm 4 Parameterized 4-list sum algorithm based on Howgrave–Graham–Joux [HJ10]

Require:
$\{\mathcal{L}_i\}_{i=1}^{4}$ - Sorted lists of 2^a uniform random ℓ-bit samples.
n - Number of nullified top bits per each round.
$v \in [0, a]$ - Parameter.
Output: $\mathcal{L}'$ - List of $(\ell - n)$-bit samples.

1. For each $c \in [0, 2^v)$:
   a. Look for pairs $(x_1, x_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ such that $\texttt{MSB}_a(x_1 + x_2) = c$. Store the expected number of $2^{2a-a} = 2^a$ output sums $x_1 + x_2$ in a new sorted list $\mathcal{L}'_1$. Do the same for $\mathcal{L}_3$ and $\mathcal{L}_4$ to build the sorted list $\mathcal{L}'_2$.
   b. Look for pairs $(x'_1, x'_2) \in \mathcal{L}'_1 \times \mathcal{L}'_2$ such that $\texttt{MSB}_n(|x'_1 - x'_2|) = 0$. Store the expected number of $2^{2a-(n-a)} = 2^{3a-n}$ output sums $|x'_1 - x'_2|$ in the list $\mathcal{L}'$.
2. Output $\mathcal{L}'$ of the expected length $M' = 2^{3a+v-n}$

2.2.6 $\mathcal{K}$-list Sum Problem

We introduce a variant of the $\mathcal{K}$-list sum problem [Din19] (a sub-problem of GBP [Wag02]) instantiated over the integers. The latter part of the paper discusses the connection between this problem and Bleichenbacher's framework.

Definition 2.3 ($\mathcal{K}$-list Sum Problem)**.** *Given $\mathcal{K}$ sorted lists $\mathcal{L}_1, \ldots, \mathcal{L}_\mathcal{K}$, each of which consists of 2^a uniformly random ℓ-bit integers, the* $\mathcal{K}$-list sum problem *asks one to find a non-empty list $\mathcal{L}'$ consisting of $x' = \sum_{i=1}^{\mathcal{K}} \omega_i x_i$, where $\mathcal{K}$-tuples $(x_1, \ldots, x_\mathcal{K}) \in \mathcal{L}_1 \times \ldots \times \mathcal{L}_\mathcal{K}$ and $(\omega_1, \ldots, \omega_\mathcal{K}) \in \{-1, 0, 1\}^\mathcal{K}$ satisfy $\texttt{MSB}_n(x') = 0$ for some target parameter $n \leq \ell$.*

Algorithm 4 is an instance of the $\mathcal{K}$-list sum algorithm for $\mathcal{K} = 4$. This is essentially a parameterized variant of the Howgrave–Graham–Joux [HJ10], which we analyze by extending Dinur's framework [Din19] in Section 2.4.

2.3 Timing Attacks on Montgomery Ladder

An implementation of the Montgomery ladder must be built on top of a constant-time implementation of the group law to enjoy its side-channel resistance guarantees. Any minor deviation in the number of field operations or memory access pattern in the group law can leak information about which of the two branches of a certain iteration are being evaluated, which further leaks information about the key bit. In this work, we exploit a vulnerability in the way the Montgomery ladder is prepared (line 1 of Algorithm 1), by observing that implementations employing projective coordinates will have accumulators R_0 in affine coordinates in which input point P is typically given; and R_1 in projective coordinates after a point doubling is performed. This coordinate mismatch allows the attacker to mount a cache-timing attack against the first iteration of the ladder, revealing the second MSB of the scalar. We found the issue in the popular OpenSSL cryptographic library, and in the research-oriented RELIC toolkit [A+], both apparently caused by attempting to implement a constant-time ladder on top of a group law optimized for special cases. This generality motivated us to name the discovered vulnerability under the moniker *LadderLeak*.

2.3.1 Cache-timing vulnerabilities in OpenSSL's implementation

OpenSSL contains multiple implementations of the Montgomery ladder in its codebase, depending on the choice of curve and field, so we split the discussion based on the choice of underlying field.

Binary curves For curves defined over $\mathbb{F}_{2^m}$, OpenSSL employs the well-known López-Dahab scalar multiplication algorithm [LD99], which amounts to the Montgomery ladder over points represented in López-Dahab coordinates ($x = X/Z, y = Y/Z^2$). Parameters affected are SECG curve `sect163r1`; and NIST curves B-283, K-283, K-409 and B-571 (i.e. binary curves with group order slightly below the power of two) in versions `1.0.2u` and `1.1.0l`. The latest `1.1.1` branch is not affected due to a unified and protected implementation of the ladder.

The Montgomery ladder is implemented in function `ec_GF2m _montgomery_point _multiply()` in file `crypto/ec/ec2_mult.c`. The function computes scalar multiplication $[k]P$ for fixed-length scalar k and input point $P = (x, y)$. The ladder starts by initializing two points $(X_1, Z_1) = (x, 1)$ and $(X_2, Z_2) = [2]P = (x^4 + b, x^2)$. The first loop iteration follows after a conditional swap function that exchanges these two points based on the value of the second MSB. The first function to be called within the first iteration is `gf2m_Madd()` for point addition, which starts by multiplying by value Z_1. However, since the finite field arithmetic is not implemented in constant-time for binary fields, there is a timing difference between multiplying by (1) or (x^2), since modular reduction is only needed in the latter case. In particular, a modular reduction will be computed when $Z_1 = x^2$ after the conditional swap. This happens when the second MSB is 1 because the conditional swap effectively swapped the two sets of values. A cache-timing attack can then monitor when the modular reduction code is called to reduce a non-trivial intermediate multiplication result.

Prime curves In curves defined over $\mathbb{F}_p$ for large prime p, OpenSSL `1.0.2u` employs the Montgomery ladder when precomputation is turned off, a scenario prevalent in practice since precomputation must be manually turned on for a certain point (typically a fixed generator) [TuHGB18]. Parameters affected are NIST curves P-192, P-224, P-384 and P-521; and SECG curves `secp192k1` and `secp256k1` (i.e. prime curves with group order slightly below the power of two). Curve P-256 is affected in principle, but OpenSSL has customized code enabled by default at build time. Note that `secp256k1` refers to the curve adopted for signing Bitcoin transactions with ECDSA.

In this case, OpenSSL implements the Montgomery ladder by using optimized formulas for elliptic curve arithmetic in the Weierstrass model. The ladder is implemented in `ec_mul_consttime()` within `/crypto/ec/ec_mult.c`, but which does not run in constant-time from a cache perspective, despite the naming. The ladder starts by initializing two accumulators $R_0 = P$ (in affine coordinates) and $R_1 = 2P$ (in projective coordinates). The first loop iteration is non-trivial and computes a point addition and a point doubling after a conditional swap. Depending on the key bit, the conditional swap is effective and only one point will remain stored in projective coordinates. Both the point addition and point doubling functions have optimizations in place for mixed addition, and the Z coordinate of the input point can be detected for the point doubling case implemented in function `ec_GFp_simple_dbl()`. When the input point for the doubling function is in affine coordinates, a field multiplication by Z is replaced by a faster call to `BN_copy()`. This happens when the two accumulators are not swapped in the ladder, which means that

point R_0 in affine coordinates is doubled and the second MSB is 0. The timing difference is very small, but can be detected with a cache-timing attack.

2.3.2 Implementation of the attacks

We implemented cache-timing attacks using Flush+Reload from the `FR-trace` program available in the Mastik side-channel analysis toolkit [Yar16]. We targeted OpenSSL by running the command-line signature computation in two Broadwell CPUs with models Core i7-5500U and i7-3520M clocked at 2.4GHz and 2.9GHz, respectively, and TurboBoost disabled. OpenSSL was built using a standard configuration with debugging symbols and optimizations enabled. Targeted parameters were at lowest security in each class: `sect163r1` for the binary and P-192/`secp192k1` for the prime case. Although the observed timing difference was very small in both cases, we managed to amplify it using performance degradation [ABF+16]: multiple threads running in the background penalize the targeted pieces of code (modular reduction in the binary case and `BN_copy()` in the prime case) by constantly evicting their addresses from the cache. The timing difference for computing the first iteration of the ladder was amplified to around 100,000 and 15,000 cycles for the binary and prime case, respectively. Amplifying the timing difference made it feasible to detect the second MSB with high probability: around 99% for curves `sect163r1` and P-192. We configured the slot, or the time between two consecutive probings by the Flush+Reload thread, to 5,000 cycles to obtain finer granularity.

In the binary case, the detection strategy consisted of first locating in the traces a cache-hit matching the execution of the first field multiplication by Z_1 in `gf2m_Madd()` at the beginning of the first ladder iteration, and then looking for the next cache-hit matching the second field multiplication which marks the end of the first. If the number of slots between the two was above 15, this means a timing difference of at least 75,000 cycles. The first version of the attack achieved 97.3% precision, which was later improved. When running the attack against 10,000 signature computations, we were able to correctly detect 2,735 signatures with second MSB 1 and only 27 false positives, amounting to a precision of 99.00%. Sample traces illustrating the strategy can be found in Figure 2.1.

In the prime case, the detection strategy consisted of looking for the first ladder iteration by locating in the traces for a cache-hit matching the execution of `ec_GFp_simple_dbl()` and then counting the number of consecutive cache-hits matching the execution of `BN_copy()`. If the next two slots had cache-hits for the latter, this means that the copy took around 3 slots, or 15,000 cycles. When running the attack against 10,000 signature computations, we were able to correctly detect 2,343 signatures with second MSB 0 and only 12 false positives, amounting to a precision of 99.53%. Sample traces illustrating the strategy can be found in Figure 2.2.

2.3.3 Translating Nonce Leakages to the Hidden Number Problem Instance

In OpenSSL, the nonce $k \in \{1, \ldots, q-1\}$ is rewritten to be either $\hat{k} = k + q$ or $\hat{k} = k + 2q$ before passed to a scalar multiplication algorithm, so that the resulting $\hat{k}$ has the fixed bit length. This is to countermeasure the remote timing attacks of Brumley and Tuveri [BT11]. For the curves with group order slightly below the power of 2 (denoted by $q = 2^\ell - \delta$), it holds that $\hat{k} = k + q$ except with negligible probability. Our *LadderLeak* attack detects the second MSB of $\hat{k}$ and we argue that it coincides with the first MSB of k with

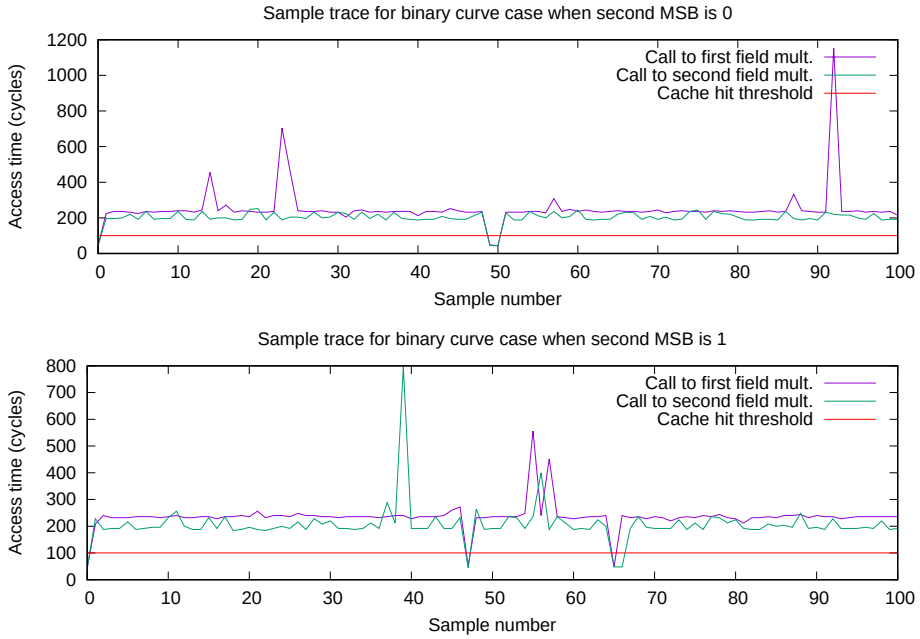

Figure 2.1: Pattern in traces collected by `FR-trace` for the binary curve case. Cache accesses are considered hits when below the default threshold of 100 cycles. The cache hits correspond to executions of the two first field multiplications inside point addition, the first by Z_1. When the second MSB is 0 and $Z_1 = 1$ in the first trace above, there is no modular reduction, hence the two first field multiplications in point addition quickly follow in succession. When the second MSB is 1 and $Z_1 = x^2$ in the second trace, performance degradation penalizes modular reduction and the time between the two field multiplications grows much larger.

overwhelming probability. Let us denote the ℓ-th bit of k (resp. $\hat{k}$) by k_ℓ (resp. $\hat{k}_\ell$). Then $\Pr[k_\ell \neq \hat{k}_\ell] < \Pr[k_\ell = 0 \wedge k < \delta] + \Pr[k_\ell = 1 \wedge k < \delta + 2^{\ell-1}]$ since the ℓ-th bit of q is 1 and k_ℓ gets flipped only if there's no carry from the lower bits in the addition $k + q$. It is easy to see that the right-hand side of the above inequality is negligibly small if δ is negligibly smaller than q. Therefore, putting together with the usual conversion in Section 2.2.3 we have obtained HNP instances with error rate at most $\epsilon = 0.01$ for P-192 and $\epsilon = 0.027$ for `sect163r1`.

2.4 Improved Analysis of Bleichenbacher's Attack

2.4.1 Unified Time–Space–Data Tradeoffs

The FFT-based approach to the HNP typically requires significant amount of input signatures, compared to lattice-based attacks. The sort-and-difference method attempted by Aranha et al. [AFG+14], for instance, required 2^{33} input signatures to break 160-bit ECDSA with 1-bit bias. We could of course take the same approach to exploit the leakage of `sect163r1` from the previous section, but collecting over 8 million signatures via cache attacks doesn't seem very easy in practice. Takahashi, Tibouchi and Abe [TTA18a] took much more space-efficient approach by making use of Howgrave–Graham and Joux's (HGJ) knapsack solver [HJ10]. They also provide "lower bounds" for the required amount

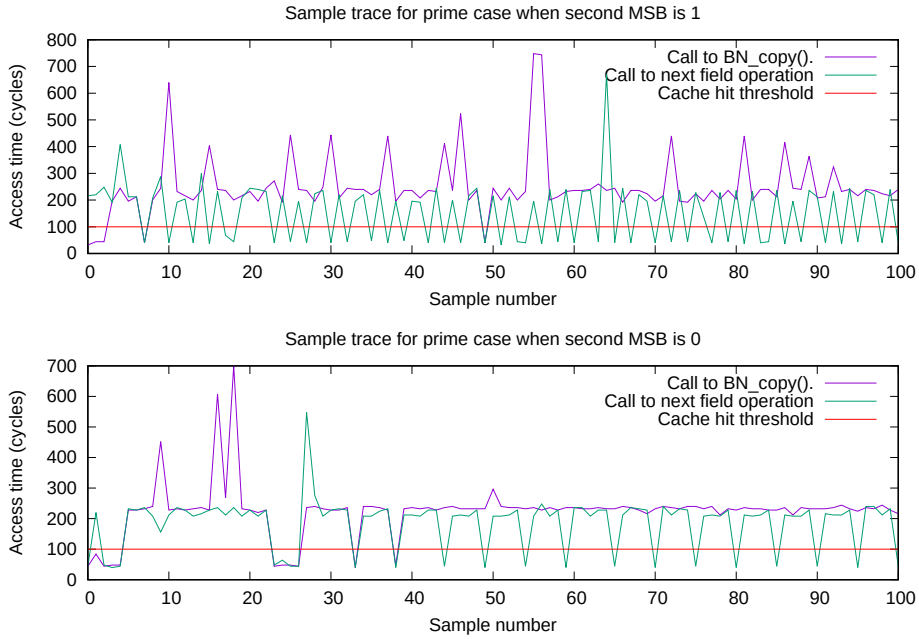

Figure 2.2: Pattern in traces collected by `FR-trace` for the prime curve case. Cache accesses are again considered hits when below the default threshold of 100 cycles. The cache hits correspond to the time a `BN_copy()` operation inside point doubling takes to complete under performance degradation. When the second MSB is 1 in the first trace, `BN_copy()` is not called inside point doubling, but the cache line containing the function call and next field operation is brought to the cache. When the second MSB is 0, `BN_copy()` is actually called and takes longer to complete due to performance degradation. The pattern is visible between slots 20 and 30.

of input samples to attack given signature parameters and bit biases. However, their lower bound formula implicitly relies on two artificial assumptions: (1) the number of input and output samples, space complexity, and FFT table size are all equal (i.e. $M = M' = L_{\mathrm{FFT}}$ in Algorithm 3), and (2) the number of collided bits to be found by the HGJ algorithm is fixed to some constant. Such assumptions do help stabilizing time and space complexities throughout the entire attack, but at the same time seem to sacrifice the true potential of applying the HGJ algorithm. In [FGMN16], Fouque et al. briefly mentioned that an algorithm for the GBP helps reducing the number of signatures required in Bleichenbacher-style attacks by initially amplifying the amount of samples, although their analysis was neither detailed nor general. In fact, the HGJ-like algorithm implemented in [TTA18a] can be regarded as an instance of the generalized birthday algorithm analyzed by Wagner [Wag02] and Dinur [Din19]. The latter in particular analyzes the time–space tradeoffs in detail, which we would like to apply and extend by introducing the third parameter, the *data complexity*. Our formulation below is motivated by a practical situation where the adversary may want to trade the "online" side-channel detection costs (i.e. data complexity) for the "offline" resources required by Bleichenbacher's attack (i.e. time and space complexities). We are now set out to address the following question.

> *For given most significant bits information in the HNP and the attacker's budget for computational resources, what would be the optimal balance between the time, memory, and input data complexities?*

2.4.1.1 Tradeoffs for Parameterized 4-list Sum Algorithm

We begin by presenting our mild generalization of Dinur's tradeoff formula (the one for the algorithm denoted by $A_{4,1}$). The main difference is that we made the number of output samples arbitrary M'.

Theorem 2.1. *For Algorithm 4, the following tradeoff holds.*

$$2^4 M' N = T M^2$$

or put differently

$$m' = 3a + v - n$$

where each parameter is defined as follows: $N = 2^n$, where n is the number of top bits to be nullified; $M = 2^m = 4 \times 2^a$ is the number of input samples, where 2^a is the length of each sublist; $M' = 2^{m'} \leq 2^{2a}$ is the number of output samples such that the top n bits are 0; $v \in [0, a]$ is a parameter deciding how many iterations of the collision search to be executed; $T = 2^t = 2^{a+v}$ is the time complexity.

Proof. For each partial target value $c \in [0, 2^v)$, Step 1.a. takes $\tilde{O}(2^a)$ time and $O(2^m)$ space to find 2^a pairs that sum to c in the top a bits, e.g. by employing the sort-merge join-like algorithm of [TTA18a]. At Step 1.b. since two pairs x'_1 and x'_2 are guaranteed to collide in the top a bits the probability that the collision occurs in the top n bits is $1/2^{n-a}$. Hence we get approximately $2^{2a}/2^{n-a} = 2^{3a-n}$ linear combinations[5]. Iterating these steps 2^v times, we get in total $M' = 2^{m'} = 2^{3a+v-n}$ samples in $\tilde{O}(2^{a+v})$ time and $O(2^m)$ space. As the algorithm goes through at most $2^a \times 2^v$ linear combinations of four it follows that $2^{m'} \leq 2^{2a}$. □

The above tradeoff gives more flexibility to the sample amplification; as the formula implies one could amplify the number of input samples to at most 2^{2a} by reducing nullified bits, or by increasing time or memory complexity. This is in particular important in Bleichenbacher's framework, since it allows us to carefully coordinate the number of output samples so that the noise floor is sufficiently smaller than the peak.

2.4.1.2 Integration with Bleichenbacher and Linear Programming

We now integrate the above basic tradeoff formula with two crucial constraints for Bleichenbacher's attack to work; namely, smallness and sparsity of the output linear combinations. In Bleichenbacher's attack, the adversary could repeat the 4-list sum algorithm for r rounds to find small linear combinations of 4^r integers below certain budget parameter for the FFT table, $L_{\mathrm{FFT}} = 2^{\ell_{\mathrm{FFT}}}$, so that the computation of FFT becomes tractable. Hence we rewrite the tradeoff formula for each round $i = 0, \ldots, r-1$ as

$$m'_i = 3a_i + v_i - n_i$$

where we define n_i, m_i, m'_i, a_i, v_i, and t_i as in Theorem 2.1. Algorithm 5 describes the iterative HGJ 4-list sum algorithm, which calls Algorithm 4 as a subroutine. Note that

[5] We remark that one could obtain slightly more solutions here thanks to the carries of additions and subtractions, when the problem is instantiated over *integers* (but not over $\mathbb{F}_2$) [TTA18a, Theorem 1]. Accordingly the tradeoff above should be adjusted by a small constant term on the right-hand side, although this of course doesn't matter asymptotically.

Algorithm 5 Iterative HGJ 4-list sum algorithm

Require:
$\mathcal{L}$ - List of $M = 4 \times 2^a$ uniform random ℓ-bit samples.
$\{n_i\}_{i=0}^{r-1}$ - Number of nullified top bits per each round.
$\{v_i\}_{i=0}^{r-1}$ - Parameter where $v_i \in [0, a_i]$.
Output: $\mathcal{L}'$ - List of $(\ell - \sum_{i=0}^{r-1} n_i)$-bit samples of the length 2^{m_r}.

1. Let $a_0 = a$.
2. For each $i = 0, \ldots, r-1$:
   a. Divide $\mathcal{L}$ into 4 disjoint lists $\mathcal{L}_1, \ldots, \mathcal{L}_4$ of length 2^{a_i} and sort them.
   b. Apply Algorithm 4 to $\{\mathcal{L}_i\}_{i=1}^{4}$ with parameters n_i and v_i. Obtain a single list $\mathcal{L}'$ of the expected length $2^{m_{i+1}} = 2^{3a_i + v_i - n_i}$. Let $\mathcal{L} \coloneqq \mathcal{L}'$.
3. Output $\mathcal{L}'$.

Table 2.2: Linear programming problems based on the iterative HGJ 4-list sum algorithm (Algorithm 5). Each column corresponds to the objective and constraints of linear programming problems for optimizing time, space, and data complexities, respectively. The boxed equations are the common constraints for all problems.

	Time	Space	Data
minimize	$t_0 = \ldots = t_{r-1}$	$m_0 = \ldots = m_{r-1}$	m_{in}
subject to	—	$t_i \le t_{\max}$	$t_i \le t_{\max}$
subject to	$m_i \le m_{\max}$	—	$m_i \le m_{\max}$
subject to	$m_{i+1} = 3a_i + v_i - n_i$		$i \in [0, r-1]$
	$t_i = a_i + v_i$		$i \in [0, r-1]$
	$v_i \le a_i$		$i \in [0, r-1]$
	$m_i = a_i + 2$		$i \in [0, r-1]$
	$m_{i+1} \le 2a_i$		$i \in [0, r-1]$
	$m_{\mathrm{in}} = m_0 + f$		
	$\ell \le \ell_{\mathrm{FFT}} + f + \sum_{i=0}^{r-1} n_i$		
	$m_r = 2(\log \alpha - 4^r \log(\lvert B_q(\mathbf{K})\rvert))$		

now the $2^{m'_i}$ outputs from the i-th round are used as inputs to the $(i+1)$-th round, so we have $m_{i+1} = m'_i$. Moreover, we could also incorporate a simple filtering technique that trades the initial problem size for the data complexity: given $2^{m_{\mathrm{in}}}$ uniformly random ℓ-bit samples, one could keep only $2^{m_0} = 2^{m_{\mathrm{in}} - f}$ samples below $(\ell - f)$-bit for any $f \ge 0$.

With these notations in mind, the smallness condition from Algorithm 3 is expressed as $\log h'_j \le \ell - f - \sum_{i=0}^{r-1} n_i \le \ell_{\mathrm{FFT}}$, since after r iterations the top $\sum_{i=0}^{r-1} n_i$ bits of $(\ell - f)$-bit input samples get nullified. On the other hand, recall that the peak height decays exponentially in the L_1-norm of the coefficient vectors (see Section 2.2.5), many sparse linear combinations need to be found in the end to satisfy the second condition $|B_q(\mathbf{K})|^{4^r} \gg 1/\sqrt{M'}$, where $M' = 2^{m_r}$ is the number of outputs after r rounds. By introducing a new slack variable $\alpha \ge 1$ and taking the logarithm the inequality can be converted to the equivalent equation $m_r = 2(\log \alpha - 4^r \log(|B_q(\mathbf{K})|))$. Here we remark that

the slack variable α should be determined depending on the possible largest noise value, which should be somewhat larger than the average $1/\sqrt{M'}$. This can be estimated by checking the distribution of $\{h'_j\}_{j=1}^{M'}$ after the collision search in Algorithm 3: let $\mathbf{H}'$ and $\mathbf{Z}'$ be random variables corresponding to $h'_j = \sum_j \omega_{i,j} h_i$ and $z'_j = \sum_j \omega_{i,j} z_i$, and hence let $\mathbf{K}' = \mathbf{Z}' + \mathsf{sk}\mathbf{H}'$. Since Bleichenbacher's attack should detect the peak at a candidate point within $q/(2L_{\mathrm{FFT}})$ distance from the actual secret sk, all the modular biases of incorrect guess $\mathbf{K}'_x = \mathbf{Z}' + (\mathsf{sk} \pm x)\mathbf{H}' \mod q$ for $x \in [q/(2L_{\mathrm{FFT}}), q/2)$ are noise. Thus the largest noise value is $\max_x |B_q(\mathbf{K}'_x)| = \max_x(|B_q(\mathbf{K}')| \cdot |B_q(x\mathbf{H}')|)$ (due to Lemma 1 of [DHMP14]), which relies on the distribution that $\mathbf{H}'$ follows. For each concrete collision search algorithm one could experimentally find the maximum value of $|B_q(\mathbf{K}'_x)|$, and therefore can choose the appropriate α to make sure that the bias peak is larger than that. For instance, for two rounds of the iterative HGJ we observed $\max_x |B_q(\mathbf{K}'_x)| \approx 5/\sqrt{M'}$. In [ANT+20b] we discuss the estimation of noise floor in a more formal fashion.

Putting together, we obtain the unified tradeoffs in the form of linear programming problem, summarized in Table 2.2. For instance, to optimize the data complexity the goal of linear programming is to minimize the (logarithm of) number of inputs m_{in} while the problem receives budget parameters $t_{\max}, m_{\max}, \ell_{\mathrm{FFT}}$, slack parameter α, and estimated bias $B_q(\mathbf{K})$ as fixed constants. We can further iteratively solve the linear programming over choices of r to find the optimal number of rounds that leads to the best result. For small number of bit biases like less than 4-bit biases, r is at most 5, so we can efficiently find the optimal parameters. We present in Figs. 2.3 and 2.4 the optimal time and data complexities for attacking 1-bit biased HNP, with different FFT table sizes and max memory bounds. These results are obtained by solving the linear programming problems with our SageMath [TheYY] script available in our GitHub repository. More tradeoff graphs in case of few bits nonce leakages are also given in [ANT+20b].

One caveat is, that simply iterating Algorithm 4 r rounds does not necessarily guarantee the 4^r sums in the resulting list; the same element in the original list may be used more than once in a single linear combinations of 4^r when the output list of i-th round is used as input to the $i+1$-th round. This would violate the coefficient constraints of the collision search phase required in Algorithm 3, since due to the [DHMP14, Lemma 1.d.] if $\mathbf{K}$ follows the uniform distribution over $[0, q/2^l)$ then the bias peak cancels out, i.e. $|B_q(2^l\mathbf{K})| = 0$. To circumvent the issue one could alternatively use the "layered" HGJ algorithm due to Dinur [Din19], of which we present a generalized variant in [ANT+20b] together with its tradeoff linear programming problems in [ANT+20b]. This way, the input list is first divided into 4^r sub-lists and the algorithm guarantees to find linear combinations composed of single element per each sub-list, while we observe that the concrete complexities for attacking our OpenSSL targets are worse than the iterative HGJ. In practice, a few iterations of HGJ algorithm outputs a negligible fraction of such undesirable linear combinations, and hence the actual bias peak is only slightly lower than the estimated $|B_q(\mathbf{K})|^{4^r}$. This heuristic was also implicitly exploited by [TTA18a] and we chose to make use of Algorithm 5 for the better performance in the attack experiments.

Finally, we remark that our approach is generic, allowing to integrate in principle any $\mathcal{K}$-list integer sum algorithms to establish a similar time–space–data tradeoff formula. In [ANT+20b], we present more linear programming problems derived from other $\mathcal{K}$-list sum algorithms, such as the 16-list sum due to Becker, Coron and Joux (BCJ) [BCJ11] and its multi-layer variant by Dinur [Din19]. For the specific HNP instances related to our attack on OpenSSL, these $\mathcal{K}$-list sum algorithms provide slightly worse complexities than

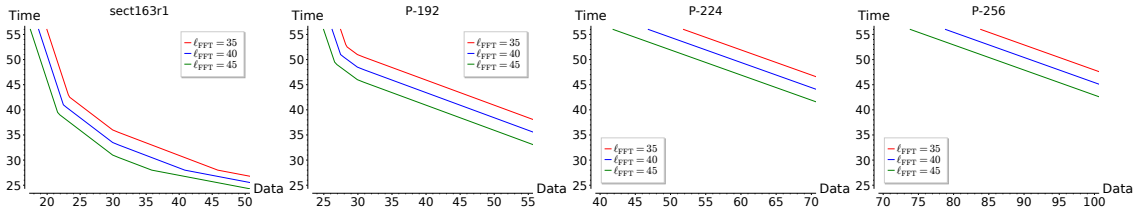

Figure 2.3: Time–Data tradeoffs where $m_{\max} = 30$, nonce k is 1-bit biased, slack parameter $\alpha = 8$ and the number of rounds $r = 2$.

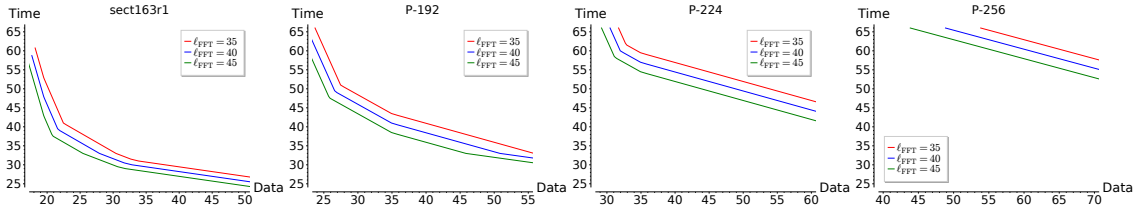

Figure 2.4: Time–Data tradeoffs where $m_{\max} = 35$, nonce k is 1-bit biased, slack parameter $\alpha = 8$ and the number of rounds $r = 2$.

the iterative HGJ. We leave for future work the discovery of parameter ranges where those alternatives perform better, as well as the adaptation of further list sum algorithms.

2.4.2 Bias Function in Presence of Misdetection

All previous FFT-based attack papers [DHMP14, AFG+14, TTA18a] only considered the idealized setting where input HNP samples have no errors in MSB information (corresponding to $\epsilon = 0$ in Section 2.2.3). As observed in Section 2.3, however, this is usually not the case in practice since the side-channel detection is not 100 percent accurate. This motivates us to consider the behavior of the bias function on non-uniformly biased samples. Below we show how to concretely calculate biases when there are ϵ errors in the input. For instance, our cache timing attack yields HNP samples with $\epsilon = 0.01$ for P-192 (resp. $\epsilon = 0.027$ for `sect163r1`), and the present lemma gives $|B_q(\mathbf{K})| = (1-2\epsilon)|B_q(\mathbf{K}_0)| \approx 0.98 \times 0.637$ (resp. $|B_q(\mathbf{K})| = (1-2\epsilon)|B_q(\mathbf{K}_1)| \approx 0.946 \times 0.637$). Note that the extreme case where $\epsilon = 1/2$ simply means that the samples are not biased at all, and therefore $|B_q(\mathbf{K})|$ degenerates to 0. This also matches the intuition; when the attacker gains no side-channel information about nonces it should be information theoretically impossible to solve the HNP (except with some auxiliary information like the knowledge of public key corresponding to the secret).

Lemma 2.1. *For $b \in \{0,1\}$, any $\epsilon \in [0, 1/2]$ and even integer $q > 0$ the following holds. Let $\mathbf{K}$ be a random variable following the weighted uniform distribution over $\mathbb{Z}_q$ below.*

$$\Pr[\mathbf{K} = k_i] = (1-b) \cdot \frac{1-\epsilon}{q/2} + b \cdot \frac{\epsilon}{q/2} \text{ if } 0 \le k_i < q/2$$
$$\Pr[\mathbf{K} = k_i] = b \cdot \frac{1-\epsilon}{q/2} + (1-b) \cdot \frac{\epsilon}{q/2} \text{ if } q/2 \le k_i < q$$

Then the modular bias of $\mathbf{K}$ is

$$B_q(\mathbf{K}) = (1-2\epsilon) B_q(\mathbf{K}_b)$$

where $\mathbf{K}_b$ *follows the uniform distributions over* $[0 + bq/2, q/2 + bq/2)$.

Proof. We prove the case for $b = 0$. The other case holds by symmetry. By Definition 2.2 the bias for $\mathbf{K}$ is rewritten as follows due to the law of the unconscious statistician.

$$\begin{aligned}
B_q(\mathbf{K}) =& \mathrm{E}[e^{(2\pi\mathbf{K}/q)\mathrm{i}}] = \sum_{k_i \in \mathbb{Z}_q} e^{(2\pi k_i/q)\mathrm{i}} \cdot \Pr[\mathbf{K} = k_i] \\
=& \frac{1-\epsilon}{q/2} \sum_{k_i \in [0,q/2)} e^{(2\pi k_i/q)\mathrm{i}} + \frac{\epsilon}{q/2} \sum_{k_i \in [q/2,q)} e^{(2\pi k_i/q)\mathrm{i}} \\
=& \frac{1-\epsilon}{q/2} \sum_{k_i \in [0,q/2)} e^{(2\pi k_i/q)\mathrm{i}} + \frac{\epsilon}{q/2} \sum_{k'_i \in [0,q/2)} e^{(2\pi (k'_i+q/2)/q)\mathrm{i}} \\
=& \frac{1-\epsilon}{q/2} \sum_{k_i \in [0,q/2)} e^{(2\pi k_i/q)\mathrm{i}} - \frac{\epsilon}{q/2} \sum_{k'_i \in [0,q/2)} e^{(2\pi k'_i/q)\mathrm{i}} \\
=& \frac{1-2\epsilon}{q/2} \sum_{k_i \in [0,q/2)} e^{(2\pi k_i/q)\mathrm{i}} = (1-2\epsilon)\mathrm{E}[e^{(2\pi \mathbf{K}_0/q)\mathrm{i}}]
\end{aligned}$$

where $k'_i \coloneqq k_i - q/2$ and we used $e^{(2\pi(k'_i+\pi)/q)\mathrm{i}} = -e^{(2\pi k'_i/q)\mathrm{i}}$. □

For brevity, we omit an almost identical result for odd q. We remark that if q is odd then there is a tiny additive error of order $1/q$. In practice, such an error is negligible since q is always significantly large for the actual HNP instances, and we experimentally confirmed that the bias peak for odd q behaves as if q was even.

2.4.3 Concrete Parameters to Attack OpenSSL

`sect163r1` To showcase the power of our tradeoff formula we describe how to concretely choose the optimal parameters to exploit error-prone 1-bit leakages from OpenSSL ECDSA. For `sect163r1`, the attacker would be able to obtain $\ell = 162$-bit HNP samples with error rate at most $\epsilon = 0.027$ due to our cache timing side-channel analysis. By Lemma 2.1 the modular bias is estimated as $|B_q(\mathbf{K})| \approx 0.602$. Suppose the attacker's computational budget is $t_{\max} = 44$, $m_{\max} = 29$, $\ell_{\mathrm{FFT}} = 34$ and let the slack variable $\alpha = 8$. We show in the next section that such computational facilities are relatively modest in practice. If the attacker's goal is to minimize the number of input samples m_{in}, then by solving the linear programming for the data complexity optimization we obtain the solution $m_{\mathrm{in}} = 24$. Our solver script gives all intermediate attack parameters, suggesting the following attack strategy that amplifies the number of samples by 2^5 during the first round.

- The first round generates $2^{m_1} = 2^{29}$ samples with top $n_0 = 59$ bits nullified via Algorithm 4 in time $2^{t_0} = 2^{a_0+v_0} = 2^{22+22} = 2^{44}$, given $2^{m_0} = 2^{m_{\mathrm{in}}} = 2^{24}$ input samples.
- The second round generates $2^{m_2} = 2^{29}$ samples with top $n_1 = 69$ bits nullified via Algorithm 4 in time $2^{t_1} = 2^{a_1+v_1} = 2^{27+17} = 2^{44}$, given $2^{m_1} = 2^{29}$ input samples.
- After $r = 2$ rounds of the collision search phase, the bias computation phase does the FFT of table size $2^{\ell_{\mathrm{FFT}}} = 2^{\ell - n_0 - n_1} = 2^{34}$, expecting to find the peak of height $|B_q(\mathbf{K})|^{4^2} = \alpha/\sqrt{2^{m_2}} \approx 0.0003$ and then recover the top 34 bits of sk.

Notice that the required number of input signatures is now significantly lower than what would have been derived from the previous published works. For example, the implementation of Aranha et al. [AFG+14] would require over 2^{33} input samples in our setting. The lower bound formula found in [TTA18a] with the same slack parameter would yield 2^{29} signatures as input for 2 rounds of the HGJ algorithm. Surprisingly, the best previous attack record dates back to 2005, in which Bleichenbacher informally claimed to break 160-bit DSA given only 2^{24} signatures [Ble05] although no details have been ever explained to date. Since the claimed number of samples does match our result, our parameter choice above may explain what he conducted back then. Moreover, all these works only considered the setting where HNP samples come without any errors in the MSB information. In such an ideal case, our tradeoff formula actually allows to mount the attack given only 2^{23} samples with almost the same time and space complexity. [ANT+20b] describes how it can be achieved in detail.

NIST P-192 Attacking the $\ell = 192$-bit HNP would be much more costly in terms of time complexity, and the attacker would be likely to want to minimize the time. Hence we now present a solution to the linear programming problem for the optimal time complexity. Note that our cache attack had the error rate at most $\epsilon = 0.01$, so the estimated bias peak is slightly more evident than the one for `sect163r1`. Suppose the attacker is given $2^{m_{\text{in}}} = 2^{35}$ samples as input and its computational budget is $m_{\max} = 29$, $\ell_{\text{FFT}} = 37$ and now let us set the slack variable $\alpha = 16$ to observe more prominent peak than before.

- As a preprocessing phase, filter $f = 6$ bits to collect $2^{m_0} = 2^{m_{\text{in}}-f} = 2^{29}$ samples such that $h < 2^{\ell-f} = 2^{186}$ holds.
- The first round generates $2^{m_1} = 29$ samples with top $n_0 = 75$ bits nullified via Algorithm 4 in time $2^{t_0} = 2^{a_0+v_0} = 2^{27+23} = 2^{50}$, given $2^{m_0} = 2^{29}$ input samples.
- The second round generates $2^{m_2} = 30$ samples with top $n_1 = 74$ bits nullified via Algorithm 4 in time $2^{t_1} = 2^{a_1+v_1} = 2^{27+23} = 2^{50}$, given $2^{m_1} = 2^{29}$ input samples.
- After $r = 2$ rounds of the collision search phase, the bias computation phase does the FFT of table size $2^{\ell_{\text{FFT}}} = 2^{\ell-f-n_0-n_1} = 2^{37}$, expecting to find the peak of height $|B_q(\mathbf{K})|^{4^2} = \alpha/\sqrt{2^{m_2}} \approx 0.0005$ and then recover the top 37 bits of sk.

Such optimized time complexity allowed us to practically solve the previously unbroken 192-bit HNP even with erroneous 1-bit leakage. Moreover, if we assume error-free input then only 2^{29} samples are needed to solve the instance with almost the same computational cost as described in [ANT+20b]. We remark that the lower bound formula of [TTA18a] with the same slack parameter, filtering bits and modular bias would yield almost the same number of signatures as input. However, their non-parameterized HGJ algorithm exhaustively looks at all bit patterns in top a bits and tries to find collisions there (which can be seen as a special case of $A_{4,2^m}$ algorithm in Dinur's framework by fixing the parameter $v = a$). The resulting algorithm would thus run in quadratic time, leading to a much worse time complexity of around 2^{56}.

2.5 Experimental Results

2.5.1 Optimized Parallel Implementations

We implemented Bleichenbacher's attack instantiated with Algorithm 5 as a collision search method. Our MPI-based parallel implementation is built upon the public available code

Table 2.3: Summary of the experimental results. The "Thread" columns are of the format #shared-memory threads × #distributed-memory nodes. The "Recovered MSBs" were computed with respect to the relative error from the actual secret sk, i.e. $\lfloor \ell - \log |\mathsf{sk} - w| \rfloor$, where w is an estimated secret key and ℓ is the bit-length of group size. We remark that the large body of memory consumption is due to the parallelization overhead, and in fact, the per-thread RAM usages were below 6GB in the collision search phase and 32GB in FFT, respectively.

Target	Facility	Cost	Error rate	Input	Output	Thread (Collision)	Time (Collision)	RAM (Collision)	L_{FFT}	Thread (FFT)	Time (FFT)	RAM (FFT)	Peak Height	Max Noise	Recovered MSBs
NIST P-192	AWS EC2	$16,429	0	2^{29}	2^{27}	96×24	113h	492GB	2^{38}	128×2	0.5h	4TB	7.28×10^{-4}	4.48×10^{-4}	39
NIST P-192	AWS EC2	$7,870	0.010	2^{35}	2^{30}	96×24	52h	492GB	2^{37}	1	12h	4TB	5.04×10^{-4}	1.55×10^{-4}	39
`sect163r1`	Cluster	–	0	2^{23}	2^{27}	16×16	7h	80GB	2^{35}	8×8	1h	128GB	4.92×10^{-4}	4.29×10^{-4}	36
`sect163r1`	Workstation	–	0.027	2^{24}	2^{29}	48	42h	250GB	2^{34}	16	1h	512GB	2.82×10^{-4}	2.21×10^{-4}	35

base of Takahashi et al. [TT18, TTA18a], and we applied various optimizations to it, which we summarize below.

- Our implementation accepts the flexible configurations of attack parameters, to fully accommodate the tradeoffs observed in the previous section, while [TTA18a] only allowed to exhaustively nullify the fixed number of bits and did not support any sample amplifications.
- After the preliminary collision search phase in top a bits between two lists, we only keep the top 64 bits of linear combinations of two, instead of 128 bits as [TTA18a] did. Without losing the correctness of the algorithm, this allows us to represent samples using the standard `uint64_t` type and avoid the multiprecision integer arithmetic altogether in later collision search phase. Due to this change, both the RAM usage and cycle counts have been improved by a factor two.
- The 4-list sum algorithm requires to frequently sort the large arrays of uniformly distributed random elements (i.e. sorting of $\mathcal{L}'_1$ and $\mathcal{L}'_2$ in Algorithm 4 step 1.a.). In such a situation the radix sort usually performs better than comparison sort algorithms like the quick sort used by [TTA18a]. By utilizing the `spreadsort` function of Boost library[6] we achieved a maximum speedup factor of 1.5.
- In [AFG+14] and [TTA18a] the FFT computations were carried out in a single-node machine. To achieve scalability we utilize the distributed-memory FFT interfaces of FFTW [FJ05].

2.5.2 Attack Experiments

NIST P-192 We exploited AWS EC2 to attack two HNP instances without errors and with $\epsilon = 0.001$ error. The concrete attack parameters are described in [ANT+20b] and Section 2.4.3, respectively. To simulate the ECDSA signatures with side-channel leakage, we first obtained 2^{29} and 2^{35} signatures with (erroneous) MSB information of nonces using our custom command line interface relying on modified OpenSSL `1.0.2u` (in a way that the MSB information of k gets exposed according to the same empirical profile built on Section 2.3), and then preprocessed them to initialize the HNP samples as in Section 2.2.3. The entire signature generation took 114 CPU hours and 7282 CPU hours in total for each case, and the latter computation was parallelized. The experimental results for the first iteration of Bleichenbacher's attack are summarized in Table 2.3. For both experiments

[6] https://www.boost.org/doc/libs/1_72_0/libs/sort/doc/html/index.html

we used 24 `r5.24xlarge` on-demand instances (with 96 vCPUs for each) to carry out the collision search phase. Since the current largest memory-optimized instance in EC2 is `x1e.32xlarge` (with 4TB RAM)[7] we accordingly set the FFT table size budget $L_{\mathrm{FFT}} = 2^{38}$ using two such instances. To test both parallelized and non-parallelized FFT, we launched 2 distributed-memory nodes with 128 shared-memory threads for the former experiment, and just a single thread for the latter. As a result we were able to recover the expected number of most significant key bits. The detected peak sizes matched the estimate $|B_q(\mathbf{K})|^{16}$ with a small relative error of order 2^{-5}, and the largest noise floors were about 5 times the estimated average (i.e. $5/\sqrt{2^{m_2}}$) in both experiments.

Once the top ℓ' MSBs of sk have been found, recovering the remaining bits is fairly straightforward in Bleichenbacher's framework; one could just "re-inject" the known part of the secret to the HNP samples as $k = z + h \cdot \mathsf{sk} = (z + h \cdot \mathsf{sk}_{\mathrm{hi}} \cdot 2^{\ell-\ell'}) + h \cdot \mathsf{sk}_{\mathrm{lo}}$, where $\mathsf{sk} = \mathsf{sk}_{\mathrm{hi}} \cdot 2^{\ell-\ell'} + \mathsf{sk}_{\mathrm{lo}}$, Thus one would obtain a new set of HNP samples and apply Algorithm 3 iteratively to recover the top bits of $\mathsf{sk}_{\mathrm{lo}}$. These iterations are much more efficient than the top MSB recovery because now the collision search only needs to find the linear combinations smaller than $2^{\ell_{\mathrm{FFT}}+\ell'}$ (see [DHMP14] for more details). Following the convention from previous works we took a small security margin; assuming that only $\ell_{\mathrm{FFT}} - 4$ bits are correctly recovered for each iteration, we set the search space for the unknown $\mathsf{sk}_{\mathrm{lo}}$ to a slightly larger interval $[0, 2^{\ell'+4}]$. In our experiment, we ran Algorithm 3 in total 5 times until the top 170 bits of sk are recovered and then did the exhaustive search to find the remaining 22 bits. All but first iterations have been completed in around 6 hours using Intel Xeon E5-2670 CPU $\times 2$ (16 cores in total) with 128GB RAM.

`sect163r1` We exploited parallel cluster computing nodes (Intel Xeon E5-2670) and workstation (Intel Xeon E5-2697) to attack two HNP instances without errors and with $\epsilon = 0.0027$ error. The concrete attack parameters for the former are described in [ANT+20b] and the ones for the latter were already described in Section 2.4.3. We first generated 2^{23} and 2^{24} ECDSA signatures like in the case of P-192, which took 1.8 and 3.6 CPU hours respectively. The measured experimental results are in Table 2.3. The recovery of remaining bits was carried out as well and it took about 2 hours using the single computing node. We observed that the peak height after the collision search for `sect163r1` without error is lower than the estimated $|B_q(\mathbf{K})|^{16} \approx 7.3 \times 10^{-4}$. We conjecture that this may have been caused by the second round of HGJ applying to distributions which are no longer uniform, although the formal analysis is left for future work. Owing to our optimized implementation both attacks succeeded with relatively modest computational costs compared to previous works. Since the CPU times are below 3 months and per-thread memory usage was 32GB in both cases we can infer that the entire attack could be easily performed with a current laptop.

AWS Cost Estimates to Attack NIST P-224 and P-256 Fig. 2.4 indicates that given 2^{35} P-224 signatures and 2^{35} memory space one could complete the collision search phase in $2^{54.5}$ time and then solve the HNP by computing the FFT of size 2^{45}. Hence we can infer from the above empirical results the concrete AWS costs to break P-224 given only 1-bit nonce leaks; indeed, the entire computation for such a case could be completed by paying about $300,000 to Amazon and running 256 `x1e.32xlarge` instances for 45 days (even considering the parallelization overhead), which should be practically doable for well-funded adversaries. Breaking P-256 with 1-bit leakage remains challenging; however,

[7] https://aws.amazon.com/ec2/pricing/on-demand/

if 2 bits of leakage are available thanks to a more powerful side-channel attack, then [ANT+20b] implies that key recovery is doable with the same computational/AWS costs, given about half a million signatures.

2.6 Software Countermeasures

The main countermeasure to defend against our attack is enforcing constant-time behavior in the implementation of scalar multiplication. We discuss three options, in increasing implementation complexity: Z-coordinate randomization, constant-time evaluation of the group law and alternative scalar multiplication algorithms.

Given a high-quality entropy source also required to generate nonces for ECDSA, the countermeasure that is easiest to implement is randomization of Z-coordinates to guarantee all intermediate points in project coordinates. This is a popular countermeasure when implementing the Montgomery ladder in Curve25519, although its exact efficacy against side-channel attacks in that context is not entirely clear [DHH+15]. We further note that additional care must be taken when converting from projective to affine coordinates at the end of the computation to prevent a related attack [CAGB20].

Another potential countermeasure is to refactor the implementation to satisfy constant-time guarantees. A first option is to implement the group law in constant time using the complete formulas in [RCB16], admitting a substantial performance impact [SS19]. There are other alternatives for the scalar multiplication algorithm which do not penalize performance as much. For example, the SPA-resistance left-to-right double-and-add scalar multiplication strategy by Coron [Cor99] computes a point addition and a point doubling at every iteration, using a conditional copy to select the correct result at the end. This strategy would have comparable performance to Montgomery ladder in the Weierstrass model while being conceptually simpler to implement securely. Other alternatives would be implementing the ladder over co-Z arithmetic to remove explicit handling of Z-coordinates [Mel07], or a dedicated exception-free ladder that favors constant-time execution [SM16].

Our patches submitted as part of coordinated disclosure implement the coordinate randomization countermeasure to randomize both accumulators independently as a defense-in-depth measure, without needing to take into account how the underlying field arithmetic is implemented. We validated the effectiveness of the countermeasure in both binary and prime curves by failing to mount the same cache-timing attacks against the patched implementations, and later by runing the `dudect` dynamic analysis tool [RBV17]. Our patches illustrating the countermeasure are available in our GitHub repository, together with datasets for signature computation containing the cache-timing traces and `dudect` integration.

Following recent trends in research and practice, we recommend new applications of digital signatures to adopt more modern schemes that facilitate constant-time implementation [BDL+12].

Chapter 3

Security of Hedged Fiat-Shamir Signatures

3.1 Introduction

Deterministic Signatures and Fault Attacks Some signature schemes require a fresh, secret random value per-signature, sometimes called a nonce. Nonce misuse is a devastating security threat intrinsic to these schemes, since the signing key can be computed after as few as two different messages are signed using the same value. The vulnerability can result from either programming mistakes attempting to implement non-trivial cryptographic standards, or faulty pseudo-random number generators. After multiple real-world implementations were found to be surprisingly vulnerable to this attack [fai10, BR18] researchers and practitioners proposed deterministic signature schemes, such as EdDSA [BDL+12], as a countermeasure, in which per-signature randomness is derived from the message and secret key as a defense-in-depth mechanism. However, it has been shown that simple low-cost fault attacks during the computation of the derandomized signing operation can leak the secret key by artificially provoking nonce reuse or by corrupting computation in other ways [Bae14, Sch16, BP16, ABF+18]. Recent papers have experimentally demonstrated the feasibility of these attacks [RP17, PSS+18, SB18]. Moreover, [BP18] and [RJH+19] extended such fault attacks to exploit deterministic lattice-based signature schemes among round two candidates of the NIST Post-Quantum Cryptography Standardization Process [AASA+19], where resistance to side-channel attacks is a design goal. Despite these attacks, deterministic signature generation is still likely a positive outcome in improving security, since fault attacks are harder to mount.

Fault Resilience of Hedged Signatures In order to balance concerns of both nonce reuse and the threat of fault injection, some signature designs are advocating deriving the per-signature randomness from the secret key sk, message m, and a nonce n. The intention is to re-introduce some randomness as a countermeasure to fault injection attacks, and gracefully handle the case of poor quality randomness, to achieve a middle-ground between fully-deterministic and fully-probabilistic schemes. We call constructions following this paradigm *hedged signatures*. Despite the growing popularity of the *hedged* paradigm in practical signature schemes (such as in XEdDSA, VXEdDSA [Per16], qTESLA [BAA+19], and Picnic2 [ZCD+19]), to the best of our knowledge, there has been no attempt to formally analyze the fault resilience of hedged signatures in the literature. While the

hedged construction intuitively mitigates some fault attacks that exploit the deterministic signatures, it does add a step where faults can be injected, and it has not been shown if faults to the hedging operation allow further attacks, potentially negating the benefit. Therefore, we set out to study the following question within the provable security methodology:

To what extent are hedged signatures secure against fault attacks?

Concretely, we study fault attacks in the context of signature schemes constructed from identification schemes using the Fiat–Shamir transform [FS87]. We propose a formal model to capture the internal functioning of signature schemes constructed in the hedged paradigm, and characterize faults to investigate their impact across different steps of the signature computation.

We prove that for some types of faults, attacks are mitigated by the hedged paradigm, while for others, attacks remain possible. This provides important information when designing fault-tolerant implementations. We then apply our results to hedged EdDSA (called XEdDSA) and the Picnic2 post-quantum signature scheme [ZCD+19], both designed using the hedged construction. The XEdDSA scheme is used in the Signal protocol [CCD+17] which is in turn used by instant messaging services such as WhatsApp, Facebook Messenger and Skype.

Threat Model We consider a weaker variant of the standard adversary assumed in the fault analysis literature [JT12], who is typically capable of injecting a fault into an arbitrary number of values. Our adversary is capable of injecting a single-bit fault each time a signature is computed. We further restrict the faults to be injected at the interfaces between the typical *commit, challenge, and response* phases of Fiat–Shamir signatures, i.e., only those function inputs and outputs can be faulted. This models transient faults injected into registers or memory cells, but does not fully capture persisting faults that permanently modify values in key storage, voltage glitches to skip instructions or micro-architectural attacks to modify executed instructions (such as RowHammer and variants [KDK+14]).

We argue that, even if our model does not capture *all* possible fault attacks, it provides a meaningful abstraction of a large class of fault attacks, and thus our analysis provides an important first step towards understanding the security of hedged signatures in the presence of faults. This way, designers and implementers can focus on protecting the portions of the attack surface that are detected as *most relevant* in practice. We observe that the effects of fault attacks found in the literature targeting deterministic signatures can be essentially characterized as simple bit-tampering faults on function input/output, even though some of actual experiments cause faults during computation [BP18].

Moreover, an abstract model is needed to prove general results, and the general functions common to all Fiat–Shamir signatures are a natural candidate for abstraction.

We consider two single-bit tampering functions to set or flip individual bits, respectively: $\texttt{flip\_bit}_i(x)$ to perform a logical negation of the i-th bit of x, and $\texttt{set\_bit}_{i,b}(x)$ to set the i-th bit of x to b. This captures both stuck-at and bit-flip fault injection attacks [KSV13], introduced as data flows through the implementation. Such attacks are practically targeted at various components of the device, e.g. memory cells, processor registers, or data buses.

3.1.1 Our Contributions

A new security model for analyzing fault attacks. We establish a formal security model tailored to Fiat–Shamir type signatures (hedged, deterministic or fully probabilistic).

We survey the literature on fault attacks, showing that our model captures many practical attacks. As a first step, we abstract real-world hedged signature schemes, basing our formalization on Bellare and Tackmann's nonce-based signatures [BT16] and Bellare, Poettering and Stebila's de-randomized signatures [BPS16]. We call this security notion *unforgeability under chosen message and nonce attacks* UF-CMNA. In this security experiment, when submitting a message to the signing oracle, the adversary may also choose the random input to the *hedged extractor*, a function that derives the per-signature randomness from a nonce, the secret key, and the message.

Then we extend UF-CMNA to include resilience to fault attacks. In this security experiment the adversary plays a game similar to the UF-CMNA game, but the signing oracle also allows the attacker to specify a fault to be applied to a specific part of the signing algorithm. We identify eleven different fault types that the adversary can apply to the signing algorithm, and we denote them by by $f_0, \ldots, f_{10}$. For example, fault type f_1 applies `set_bit` or `flip_bit` to the secret key input to the hedged extractor. This notion is called *unforgeability under faults, chosen message and nonce attacks*, and is denoted F-UF-fCMNA where F is a set of fault types.

Fault resilience of hedged Fiat–Shamir signatures. We then prove that hedged Fiat–Shamir signature schemes are secure against attacks using certain fault types. Of the eleven fault types in our model, we found that the generic hedged Fiat–Shamir signature scheme is resilient to six of them (summarized in Fig. 3.1). As our model gives the attacker nearly full control of the RNG by default, our main results indicate that the hedged scheme can resist additional faults even in this (usually dire) scenario. The only constraint is that message-nonce pairs do not repeat as otherwise the scheme degenerates to a pure deterministic construction and attacks become trivial. When the underlying ID scheme has an additional property that we call *subset revealing*, the corresponding hedged signature scheme is secure against attacks that use eight of the eleven fault types.

Overall, our results give a full characterization of which fault attacks are mitigated as intended by the hedged construction, and which fault attacks remain. Our conclusion is that hedging is never worse than the deterministic construction with respect to faults, plus it has the additional benefit of hedging against poor randomness.

Fault resilience of XEdDSA and Picnic2. We use the Schnorr signature scheme throughout the paper as an example. As an application of our results, we show that hedged Schnorr resists attacks for six of the eleven fault types in our model. One implication is that the hedged scheme XEdDSA does provide better resistance to fault attacks than (deterministic) EdDSA. In particular, XEdDSA resists all fault injection attacks against EdDSA described in the literature that rely on nonce reuse without skipping nonce generation entirely [BP16, RP17, ABF+18, PSS+18, SB18]. We also show to what extent the Picnic2 signature scheme is secure against the fault attacks in our model. Because it is subset-revealing, resistance to eight of the eleven fault types is immediately established by our results for generic ID schemes. For the remaining three, we prove security for one (using specific details of Picnic2), and show attacks for the other two.

3.1.2 Related Work

To the best of our knowledge, ours is the first work considering fault attacks on hedged constructions. However, the modeling and construction of secure cryptographic schemes in the presence of faults or tampering attacks has received plenty of attention in recent

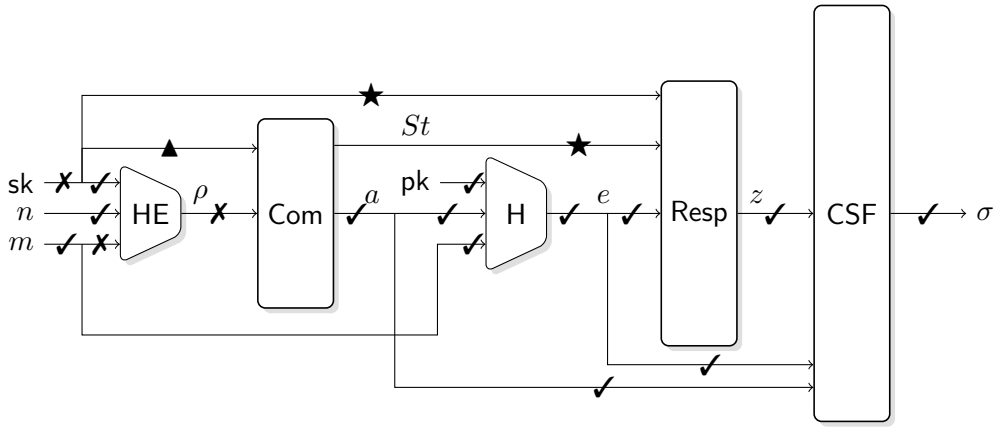

Figure 3.1: Overview of our results for hedged Fiat–Shamir type signature schemes. ✓ indicates security against 1-bit fault on the corresponding wire value, and ✗ indicates an attack or counterexample. A ★ (resp. ▲) indicates that security only holds for the schemes derived from subset-revealing ID (resp. input-delayed ID) protocols. The function components HE, Com, H, Resp, and CSF stand for hedged extractor, commitment, hash function, response, and canonical serialization function, respectively (see Sections 3.2 and 3.3 for the formal definitions).

years. We survey some of this work below. Related work on fault attacks to deterministic signature schemes is given in Section 3.2.3.

De-randomized and Hedged Constructions. Bellare and Tackmann [BT16] studied cryptography that is hedged against randomness failures. They also describe the "folklore construction", where the signing key and message to be signed are used to derive the per-signature randomness, and additional randomness may or may not be included in the derivation. Schnorr signatures with this construction have been analyzed by M'Raihi et al. [MNPV99]. A generic version of the folklore derandomization construction was proven UF-CMA secure by Bellare, Pottering and Stebila [BPS16]. Other works on hedged cryptography include [RY10] and [BBN+09, BH15, BPS17, HLC+18] when considering hedged public-key encryption in particular.

Fault Attacks and Tamper-Resilient Signatures. Tamper-resilient cryptography has received plenty of attention, both in the context of theoretical and practical cryptographic research, dating back at least to the early paper of Boneh, Demillo and Lipton [BDL97] considering fault attacks on RSA signatures (here it is noted that some attacks fail when a random padding is used, since it ensures that the same message is never signed twice). Later Coron and Mandal [CM09] proved that RSA-PSS is protected against random faults, and Barthe et al. [BDF+14] extends this to non-random faults as well. All of the above works contain examples of how randomization improves the security of signature schemes against fault attacks (in a provable way).

Other early work includes Gennaro et al. [GLM+04] that provides an early framework for proving tamper resilience, and Ishai et al. [IPSW06] which proposes generic transformation for tamper-resilient circuits. In a later work by Faust et al. [FPV11] a different and incomparable model was considered, which in particular guarantees security against tampering with *arbitrary* number of wires. We note that our model is similar to theirs since

it also considers adversaries that are allowed to flip or reset each bit in the circuit. Similar ideas are also used in practice when considering fault resilient masking (e.g. [DAN+18]).

In our model the adversary is only allowed to tamper with part of the computation. Similar limitations have been considered before in the literature to circumvent impossibility results, in particular in the so called *split-state model* [DPW18]. Several constructions have been proposed in this model including: non-malleable codes (Dziembowski, Pietrzak and Wichs [DPW18]), signature schemes (Faonio et al. [FNSV18]), and more (Liu and Lysyanskaya [LL12]).

Other related work on tamper resilient signature schemes includes [FV16, FX16, ACM+17, DFMV17]. Most of this previous work has focused on *constructing* novel tamper resilient signature schemes, or understanding the limits of tamper resilience, in theory. Instead, we focus on analyzing the tamper resilience of a popular transformation used in practice.

Related key attacks (RKA) can be seen as a special case of tampering. Bellare and Kohno [BK03] initiated the formal study of related-key attacks. Morita et al. [MSM+16] analyzed RKA security of Schnorr signatures.

Ineffective Fault Attacks (IFA) and Countermeasures. In this paper we consider not only `flip_bit` fault attacks, but also `set_bit` faults for the following reason. Clavier [Cla07] proposed *ineffective fault attacks (IFA)*, in which the adversary forces a certain intermediate bit value to be stuck at 0 or 1, and tries to recover the secret internal state by observing whether the correct output is obtained (i.e. the injected fault was ineffective). IFA is very powerful, and works even if the target algorithm contains typical countermeasures against fault attacks, such as a correctness check after redundant operations [BCN+06] and the infective countermeasure [YJ00]. IFA has been recently superseded by *statistical* ineffective fault attacks (SIFA) [DEK+18, DEG+18], that use statistical analysis to enable mounting IFA with low-precision bit-fixing, random or bit-flip faults. Daemen et al. [DDE+20] provided several practical countermeasures against SIFA, and their abstract adversarial model is close to ours in the sense that the adversaries are allowed to flip or set a single bit wire value in the circuit per query, though their security argument does not follow the provable security methodology

Concurrent Work. An independent work by Fischlin and Günther [FG20] proposes a memory fault model for digital signatures and authenticated encryption. Their main result about a generic hedged signature scheme is two-fold: it is provably secure when the nonce is fully faulted, or when the message, nonce, and hedged extractor output are all differentially faulted in each signing query. The former essentially coincides with our Lemma 3.3, but with a different proof technique. For the latter, the outcome diverges because the adversarial power in our model is different in the following ways: (1) the adversary can locally inject a fault into `sk` as a hedged extractor input, (2) the adversary can inject a bit-fixing fault, not only a bit-flip (i.e. differential) fault, (3) the adversary has nearly full control over the nonce, instead of assuming nonces are randomly generated and subject to bit flips later on, but (4) the adversary cannot inject multi-bit faults into multiple variables in a query. We additionally consider fault attacks on other various intermediate values inside the signing operation. Our treatment is then more fine-grained and successfully captures typical existing attacks on deployed deterministic schemes (like attacks that fault the challenge hash), while [FG20] does not. The upside of the generic approach in [FG20] is that the result applies to more signature schemes.

$\mathsf{Gen}(1^\kappa)$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{IGen}(1^\kappa)$
2 : **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{H}(x)$

1 : If $\mathrm{HT}[x] = \perp$:
2 : $\quad \mathrm{HT}[x] \xleftarrow{\$} D_H$
3 : **return** $\mathrm{HT}[x]$

$\mathsf{Sign}(\mathsf{sk}, m; \rho)$

1 : $(a, St) \leftarrow \mathsf{Com}(\mathsf{sk}; \rho)$
2 : $e \leftarrow \mathsf{H}(a, m, \mathsf{pk})$
3 : $z \leftarrow \mathsf{Resp}(\mathsf{sk}, e, St)$
4 : $\sigma \leftarrow \mathsf{CSF}(a, e, z)$
5 : **return** σ

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$

1 : $(a, e, z) \leftarrow \mathsf{CDF}(\sigma, \mathsf{pk})$
2 : **return** $\mathsf{V}(a, e, z, \mathsf{pk}) \stackrel{?}{=} 1$
3 : $\quad \wedge\, \mathsf{H}(a, m, \mathsf{pk}) \stackrel{?}{=} e$
4 :

Figure 3.2: The Fiat–Shamir transform applied to canonical ID with serialization CSF, to construct the signature scheme $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. The function $\mathsf{H} : \{0,1\}^* \to D_H$ is constructed with a cryptographic hash function which we model as a random oracle.

3.2 Preliminaries

Notation The notation $|\cdot|$ denotes two quantities depending on the context: $|S|$ denotes the cardinality of a set S, and $|s|$ denotes the length of a bit string s. The notation $x \xleftarrow{\$} X$ means that an element x is sampled from the set X uniformly at random. We often use the notation $[n]$ as a short hand for a set $\{1, \ldots, n\}$ where $n \in \mathbb{N}$. When we explicitly mention that an algorithm A is randomized, we use the notation $\mathsf{A}(x; \rho)$ meaning that it is executed on input x with random tape ρ. We also remark that if the lemmas/theorems are marked with "(informal)", then it means that asymptotic bounds are omitted. The full version [AOTZ19] includes more rigorous statements for all of them.

Fiat–Shamir type Signature Schemes This paper studies the robustness of Fiat–Shamir type signature schemes against fault attacks. The details of these algorithms appear in the full version. The Schnorr signature scheme [Sch91] is one of the most well-known signature schemes using the Fiat–Shamir transform, and EdDSA and XEdDSA are essentially deterministic and hedged variants of Schnorr. The Picnic2 signature scheme [ZCD+19] is constructed by applying the Fiat–Shamir transform to a three-round zero-knowledge proof system by Katz et al. [KKW18], which follows so-called "MPC-in-the-head" paradigm [IKOS07]. The hedging strategy we study in this paper is recommended in its specification.

3.2.1 Definitions

In this subsection we recall several basic definitions related to digital signatures constructed from the identification protocols. Since this paper deals with Fiat–Shamir signatures, we always assume that the signing algorithm of digital signature schemes takes some randomness as input.

We now define a three-round public-coin identification protocol, the basis of Fiat–Shamir-type signatures. The definition below essentially follows the formalization of [KLS18] unless explicitly stated.

Definition 3.1 (Canonical Identification Protocol)**.** *A* canonical identification protocol*, denoted by a tuple of algorithms* $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$*, is a three-round protocol defined as follows:*

- $\mathsf{IGen}(1^\kappa)$*, where* κ *is a security parameter, outputs a key pair* (sk, pk)*. In the context of identification protocols,* pk *and* sk *are sometimes called* statement *and* witness*. We assume that* IGen *defines a* hard-relation*, and that* pk *defines the parameters of the scheme including:* randomness space D_ρ*,* commitment space A*,* challenge space D_H *and* response space Z*.*
- Prover *invokes a committing algorithm* Com *on a secret key* sk *and randomness* $\rho \in D_\rho$ *as input, and outputs a* commitment $a \in A$ *and* state St*.*
- Verifier *samples a* challenge e *from the challenge space* $D_H \subseteq \{0,1\}^*$*.*
- Prover *executes a response algorithm* Resp *on* (sk, e, St) *to compute a* response $z \in Z \cup \{\perp\}$*, where* $\perp \notin Z$ *is a special symbol indicating failure. On top of this standard formalization, we further require that* Resp *returns* $\perp$ *whenever it receives a malformed challenge* $\tilde{e} \notin D_H$*, as such a simple sanity check is performed in most practical implementations.*
- Verifier *executes a verification algorithm* V *on* (a, e, z, pk) *as input, to output 1 (i.e. accept) or 0 (i.e. reject).*

We call a triple $(a, e, z) \in A \times D_H \times Z \cup \{\perp, \perp, \perp\}$ *a* transcript*, and it is said to be* valid *with respect to* pk *if* $\mathsf{V}(a, e, z, pk) = 1$*. We say that* ID *is* correct *if for every pair* (pk, sk) *output by* IGen*, for every* $\rho \in D_\rho$*, and for every transcript* (a, e, z) *from an honest execution of the protocol between* $\mathsf{Prover}(sk; \rho)$ *and* $\mathsf{Verifier}(pk)$*,* $\Pr[\mathsf{V}(a, e, z, pk) = 1] = 1$*.*

Remark The response algorithm in the above definition does not explicitly take a commitment a as input. We decided to do so since a is generally not required to compute z, such as in the Schnorr identification scheme and, if needed, we assume that St contains a copy of a.

The following definition is adapted from [HL10, Chapter 6]. We explicitly differentiate three flavors of the special HVZK property depending on a level of indistinguishability, following the approach found in [Gol01, Chapter 4]. Note that ϵ_{HVZK} below is equal to 0 for special *perfect* HVZK. In case HVZK is only computational, the definition should be augmented with the *auxiliary input* to enable sequential composition. In the full version [AOTZ19] we address the technicalities in detail[1].

Definition 3.2 (Special c/s/p-HVZK)**.** *Let* $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ *be a canonical identification protocol.* ID *is said to be* special computational/statistical/perfect honest-verifier zero knowledge (special c/s/p-HVZK) *if there exists a probabilistic polynomial-time simulator* $\mathcal{M}$*, which on input* pk *and* e *outputs a transcript of the form* (a, e, z) *that is computationally/statistically/perfectly indistinguishable from a real transcript between an honest prover and verifier on common input* pk*. We also denote by* ϵ_{HVZK} *the upper bound on the advantage of all probabilistic polynomial-time distinguishing algorithms.*

In our security analysis of specific hedged-signature schemes in the presence of faults we will provide a concrete bound on the min-entropy of the associated ID scheme. But here we present a useful lemma stating that the commitment message a of any secure

[1] We thank the authors of [GHHM21] for communicating this issue.

identification scheme must have high min-entropy. The lemma might be folklore but we were unable to find a reference to it, so we include it for completeness in the full version.

Lemma 3.1. *Let* ID *be a canonical identification protocol as in Definition 3.1, satisfying special-soundness and HVZK (as in Definition 3.2). Then, the min-entropy* α *of the commitment message* a *(given the public key) is at least* $\alpha = \omega(\log(\lambda))$

Definition 3.3 (Subset Revealing Identification Protocol)**.** *Let* $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ *be a canonical identification protocol. We say that* ID *is* subset revealing *if* ID *satisfies the following. 1)* St *is a set of* c *states* $\{St_1, \ldots, St_c\}$*, 2)* Resp *first derives an index set* $I \subset [c]$ *using only* e *as input, and outputs* St_i *for* $i \in I$ *as* z*, and 3)* $|St|$ *and* $|D_H|$ *are both polynomial in* κ*.*

Remark. Similar definitions were previously given by Kilian et al. [KMO90] and Chailloux [Cha19], where they make zero-knowledge or identification protocols simply reveal a subset of committed strings. Our definition generalizes their notion so that it can cover some protocols that reveal arbitrary values other than committed strings. Also notice that the Resp function of subset revealing ID schemes does not use sk at all. The above definition includes the $\mathsf{Picnic2}$ identification protocol (discussed in more detail in Section 3.6), and many classic three-round public-coin zero-knowledge proof protocols, such as the ones for graph isomorphism, Hamilton graphs, and 3-colorable graphs [GMW86]. We also emphasize that $|St|$ and $|D_H|$ need to be restricted for efficiency reasons – otherwise any identification protocol (including Schnorr) could be made subset revealing by simply precomputing (exponentially many) responses for every possible challenge and storing them in the state.

Serialization of Transcripts. For efficiency purposes, most Fiat-Shamir based signature schemes do not include the entire transcript of the identification protocol as part of the signature. Instead, redundant parts are omitted and recomputed during the verification phase. Different signature schemes omit different parts of the transcript: in some cases a is omitted and in others e is omitted. To capture this in our framework without loss of generality we introduce a *serialization* function that turns the transcript of an identification protocol into a signature.

Definition 3.4 (Canonical Serialization Function)**.** *Let* $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ *be a canonical identification protocol, and let* pk *be a public key output by* IGen*. We call a function* $\mathsf{CSF} : \{0,1\}^* \to \{0,1\}^*$ *a* canonical serialization function *if* CSF *is efficiently computable and deterministic, and satisfies the following basic properties: 1) it is* valid*, meaning that there exists a corresponding de-serialization function* CDF *which satisfies the following: for any transcript* $(a, e, z) \in A \times D_H \times Z \cup \{\bot, \bot, \bot\}$ *such that* $\mathsf{V}(a, e, z, pk) = 1$*, it holds that* $\mathsf{CDF}(\mathsf{CSF}(a, e, z), pk) = (a, e, z)$*, and 2) it is* sound with respect to invalid responses*, meaning that it returns* $\bot$ *upon receiving* $z = \bot$ *as input.*

Definition 3.5 (Fiat–Shamir Transform)**.** The Fiat–Shamir transform*, denoted by* $\mathbf{FS}$*, takes a canonical identification protocol* ID *and canonical serialization function* CSF *as input, and outputs a signature scheme* $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *defined in Fig. 3.2. For convenience, this paper refers to such schemes as* Fiat–Shamir type signature schemes*.*

Remarks By construction, it holds that if ID is correct, then $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ is a correct signature scheme. We assume ID is correct throughout the paper. In Fig. 3.2, the

verification condition may appear redundant. However, the above definition allows us to capture several variations of the Fiat–Shamir transform. For instance, a type of Fiat–Shamir transform found in some papers e.g. Ohta–Okamoto [OO98] and Abdalla et al.[AABN02] can be obtained by letting $\mathsf{CSF}(a, e, z)$ output $\sigma := (a, z)$ and letting $\mathsf{CDF}(\sigma, pk)$ call $e \leftarrow \mathsf{H}(a, m, pk)$ inside to reconstruct the whole transcript. In contrast, if ID is commitment-recoverable [KLS18], one can instantiate its serialization as follows: $\mathsf{CSF}(a, e, z)$ outputs $\sigma := (e, z)$ and $\mathsf{CDF}(\sigma, pk)$ calls $a \leftarrow \mathsf{Recover}(pk, e, z)$ inside to reconstruct the transcript.

3.2.2 Relation between UF-KOA Security and UF-CMA Security

The security notion *unforgeability against key-only attacks* (UF-KOA), is the same as UF-CMA, but with the restriction that the adversary is only given the public key, and no Sign oracle. The following result is a mild generalization of [KMP16, Lemma 3.8]: the original lemma only covers *perfect* HVZK and does not include the serialization function which we use in this work. The proof is very similar to the original one and is provided in the full version. In Section 3.4, we extend this result, showing that for some signature schemes security against key-only attacks implies security against certain fault attacks.

Lemma 3.2 (UF-KOA $\rightarrow$ UF-CMA (informal))**.** *Let* ID *be a correct canonical identification protocol and* CSF *be a canonical serialization function for* ID*. Suppose* ID *is special c/s/p-HVZK and has α-bit min-entropy. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* UF-CMA *secure in the random oracle model.*

3.2.3 Fault Attacks on Deterministic Fiat–Shamir Signatures

In recent years, several papers [BP16, RP17, ABF+18, PSS+18, SB18] presented differential fault attacks against deterministic Fiat–Shamir-type schemes. We present the conceptual overview of those previous attacks. A more detailed survey is given in the full version [AOTZ19].

Special Soundness Attack (`SSND`) This type of attack exploits the *special soundness* property of the underlying canonical identification protocol. That is, there exists an efficient algorithm that extracts the witness sk corresponding to the statement pk, given two accepting transcripts (a, e, z) and (a, e', z'), where $e \neq e'$ [Dam10]. Note in fact that it is easier to extract the secret key for an attacker than for a knowledge extractor in a proof of security, since the attacker can assume that the prover honestly follows the protocol while the special soundness property considers possibly cheating provers. `SSND` can be cheaply achieved by injecting a fault into commitment output, or hash input/output.

Large Randomness Bias Attack (`LRB`) This attack slightly modifies the randomness ρ to $\rho' = \rho + \Delta$ using, e.g. `flip_bit` fault. The attack highly relies on the deterministic property because the adversary knows that all signatures on the same message m use the same ρ, and if ρ is slightly perturbed by some sufficiently small Δ, he can find Δ with an exhaustive search. Then the adversary can recover the secret key by querying two deterministic signatures on the same message, which were computed using correlated randomness ρ and $\rho + \Delta$. `LRB` can be cheaply achieved by injecting a fault into the deterministic randomness derivation phase, or the randomness as response input.

3.3 Formal Treatment of Hedged Signatures

In this section, we give formal definitions for a hedged signature scheme and its security notion, based on Bellare–Tackmann's *nonce-based signatures* [BT16, §5] and Bellare–Poettering–Stebila's *de-randomized signatures* [BPS16, §5.1]. Then we define our new security notion for hedged Fiat–Shamir signature schemes, which guarantees resilience against 1-bit faults on function inputs/outputs.

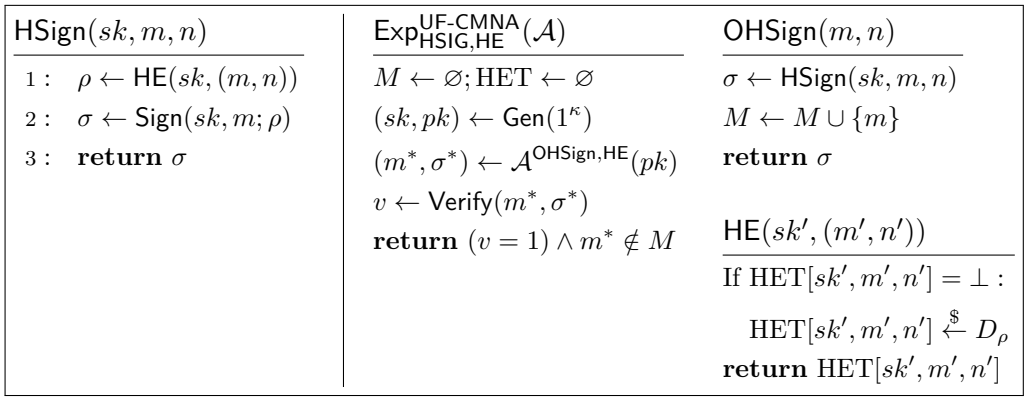

Figure 3.3: Hedged signature scheme $\mathsf{HSIG} = \mathbf{R2H}[\mathsf{SIG}, \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ and UF-CMNA experiment. Key generation and verification are unchanged.

3.3.1 Security of Hedged Signature Schemes

We now consider a simple transformation **R2H**, which converts a randomized signature scheme to a so-called "hedged" one, and its security notion UF-CMNA (unforgeability against chosen message and nonce attacks). See Fig. 3.3 for the full details. Parts of the transformation appear in the literature independently, but by combining them, we can model the concrete hedged signature schemes of interest. We now describe the differences and similarities between **R2H** and the transformations that appeared in previous works.

- On one hand, a hedged signing algorithm HSign takes a *nonce* n along with a message m, and derives the randomness $\rho \in D_\rho$ (of length ℓ_ρ bits) with a *hedged extractor* HE with $(sk, (m, n))$ as input. We do not specify how the nonces are generated here, but in practice they are the output of a pseudorandom number generator. As we will see soon, low entropy nonces do not really degrade the security of hedged signatures as long as the underlying randomized signature scheme is secure. The hedged construction we presented is essentially based on the approach taken in [BT16]. Note that HE is in practice a cryptographic hash function, that we will model as a random oracle.
- On the other hand, we use the signing key sk as the key for the hedged extractor, whereas Bellare and Tackmann used a separately generated key (which they called the "seed"), that must be stored with sk. We chose to do so in order to model concrete hedged Fiat–Shamir type schemes, such as XEdDSA and Picnic2. In fact, the security of the deterministic construction that hashes sk and m to derive ρ (with no nonce) was formally treated by Bellare–Poettering–Stebila [BPS16], and our security proof in the next section extends their result.

- Moreover, the signing oracle OHSign in our $\mathsf{UF\text{-}CMNA}$ experiment takes m and n as input adaptively chosen by the adversary $\mathcal{A}$. This can be regarded as the strongest instantiation of the oracle provided in [BT16], where nonces are derived via what they call a nonce generator (NG). Indeed, one of their results for nonce-based signatures (Theorem 5.1) does not impose any restrictions on NG, and it implicitly allows adversaries to fully control how the nonces are chosen in the signing oracle.

Now we formally define a security notion for hedged signature schemes, as a natural extension of the standard $\mathsf{UF\text{-}CMA}$ security definition. We also give a tweaked version of Theorem 4 in [BPS16], where they only consider the signing oracle that doesn't take adversarially chosen nonces. Note that Lemma 3.3 applies to *any* secure signature schemes and hence it may be of independent interest. We present a proof in the full version [AOTZ19] for completeness.

Definition 3.6 ($\mathsf{UF\text{-}CMNA}$)**.** *A hedged signature scheme* $\mathsf{HSIG} = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ *is said to be* $\mathsf{UF\text{-}CMNA}$ *secure in the random oracle model, if for any probabilistic polynomial time adversary* $\mathcal{A}$*, its advantage*

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter κ*, where* $\mathsf{Exp}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A})$ *is described in Fig. 3.3.*

Lemma 3.3 ($\mathsf{UF\text{-}CMA} \to \mathsf{UF\text{-}CMNA}$ (informal))**.** *Let* $\mathsf{SIG} \coloneqq (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a randomized digital signature scheme, and let* $\mathsf{HSIG} \coloneqq \mathbf{R2H}[\mathsf{SIG}, \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ *be the corresponding hedged signature scheme with* HE *modeled as a random oracle. If* SIG *is* $\mathsf{UF\text{-}CMA}$ *secure, then* HSIG *is* $\mathsf{UF\text{-}CMNA}$ *secure.*

3.3.2 Security of Hedged FS Type Signature Schemes Against Fault Adversaries

1-bit Transient Fault on Function Input/Output To model transient fault attackers on data flow, recall that we consider the following 1-bit tampering functions: 1) $\texttt{flip\_bit}_i(x)$, which does a logical negation of the i-th bit of x, and 2) $\texttt{set\_bit}_{i,b}(x)$, which sets the i-th bit of x to b. Using $\texttt{flip\_bit}_i(x)$ (for instance, with a random position i), we can model a typical bit-flip induced from fault injection to the memory cells, CPU register values, or data buses of the target device. Beyond faults, we also wish to capture the case in which the randomness has a 1-bit bias, which has been shown to be a serious threat for some Fiat–Shamir type signatures [AFG+14]. We can model this using $\texttt{set\_bit}_{i,b}$: when this function is applied to ρ, we can ensure that the first bit of ρ is "stuck" at zero by setting $i = 0$ and $b = 0$ to model 1-bit bias. Moreover, `set_bit` is a typical way to achieve so-called ineffective fault attacks [Cla07, DEK+18]. Our formalization covers many fault attacks found in the surveyed literature (in the full version), as they rely only on low precision faults like random bit flips of the function input or output

As a notable difference between our fault adversary model and actual attacks, some surveyed papers caused faults on several bits/bytes of function input or output when performing fault attack experiments. This is *not* to take advantage of multiple-bit faults, but rather because reliably causing a fault on a specific target memory cell is difficult in practical experiments. In fact, the attacks we classified as `SSND` and `LRB` can be achieved with uncontrolled 1-bit flip faults, and hence our model at least seems to capture the essence

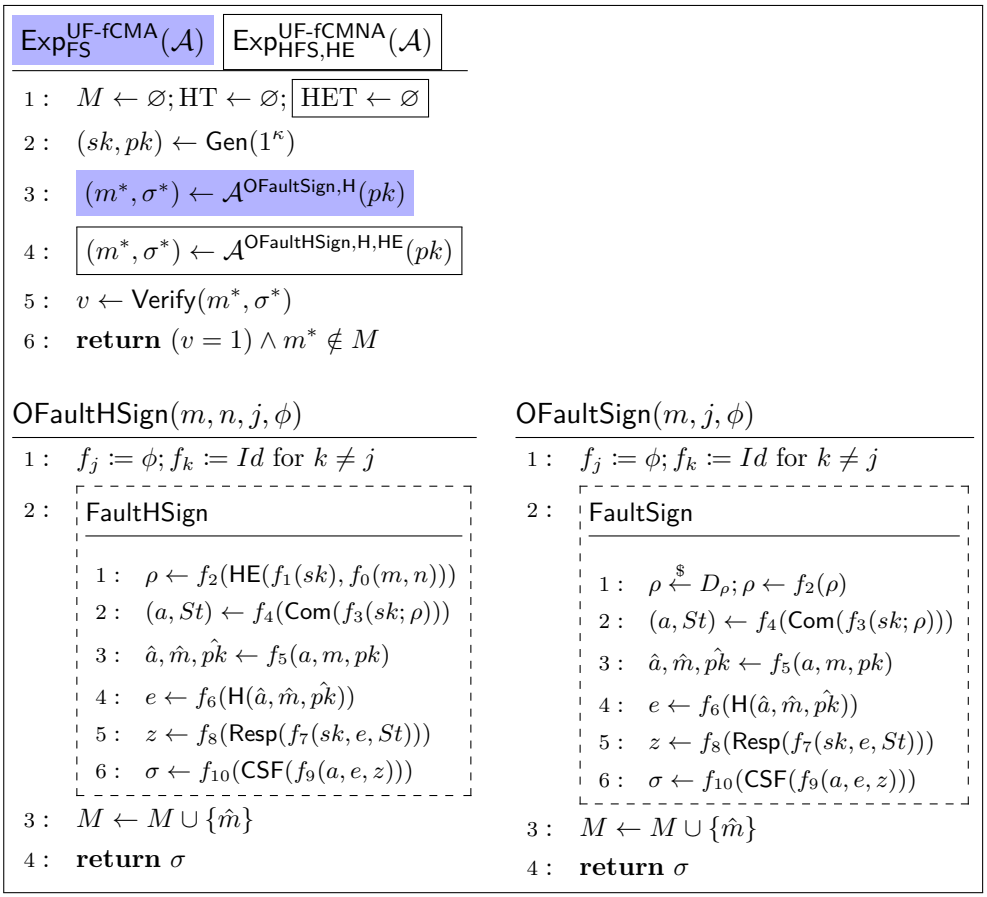

Figure 3.4: UF-fCMNA and UF-fCMA security experiments and faulty signing oracles for both hedged (HFS) and plain (FS) Fiat–Shamir signature schemes. Id stands for the identity function. The function H and HE (not shown), are the same as in Fig. 3.2 and Fig. 3.3, respectively. A dashed box indicates that the instructions inside correspond to the actual faulty signing operation.

of previous attacks exploiting the deterministic nature of signing. A natural generalization is to allow `set_bit` to work on multiple bits, for example to model word faults, or word zeroing faults. We can also model stronger attacks that are uncommon in the literature, such as setting words to arbitrary values. However, we focus on 1-bit faults in this paper as a first attempt to perform the formal analyses. We leave the security analysis against multi-bit faults for future work. In the full version, we describe some more fault attacks that are not covered by our model, to illustrate the limitations of our analysis. Each of these issues makes an interesting direction for future work.

Equipping UF-CMNA Adversaries with Faults Now we are ready to define security against fault adversaries using the above tampering functions. In Fig. 3.4, we give the modified hedged signing oracle OFaultHSign, which additionally takes a tampering function $\phi \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i, Id\}$ and $j \in [0, 10]$ as input, where Id is the identity function. This way, the adversary can specify for each query the tampering function (ϕ) as well as the target input/output position (j) within the signing operation to be faulted. For example, when $j = 6$, ϕ is applied to the output of the hash function H, and when $j = 5$ it is applied

to the input to H. The other positions are not faulted. Notice that we also allow the adversary to set $\phi := Id$ in arbitrary signing queries, so OFaultHSign includes the behavior of the non-faulty oracle OHSign as a special case. A generalization we considered but decided against, is allowing faults on multiple wire values per sign query. The combinatorial complexity of security analysis in this setting is daunting, and we did not find this to be relevant in practice, based on our survey of practical attacks.

Definition 3.7 (UF-fCMNA). *A hedged Fiat–Shamir signature scheme*

$$\mathsf{HFS} := \mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$$

is said to be F-UF-fCMNA secure, if for any probabilistic polynomial time adversary $\mathcal{A}$ who makes queries to OFaultHSign *with a fault function $f_j \in F \subseteq \{f_0, \ldots, f_{10}\}$ for each query (called F-adversary), its advantage*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter κ, where $\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A})$ is described in Fig. 3.4.

In the next section, we also use the following intermediate security notion, which essentially guarantees the security of plain randomized Fiat–Shamir signature scheme against fault adversaries.

Definition 3.8 (UF-fCMA). *A Fiat–Shamir signature scheme*

$$\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$$

is said to be F-UF-fCMA secure, if for any probabilistic polynomial time adversary $\mathcal{A}$ who makes queries to OFaultSign *with a fault function $f_j \in F \subseteq \{f_2, \ldots, f_{10}\}$ per each query (called F-adversary), its advantage*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter κ, where $\mathsf{Exp}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A})$ is described in Fig. 3.4.

Trivial Faults on the Root Input Wire Values We remark the existence of two faults on the left most input wires in Fig. 3.1, which we do not explicitly consider in our model, but its (in)security can be proven trivially. First, faulting message m before it is loaded by the signing oracle can be regarded as a situation where the adversary queries a faulty message $\hat{m}$ to begin with, since the oracle stores $\hat{m}$ in M. Hence we can just treat such a query as one to non-faulty signing oracle (OSign). Second, the adversary could easily recover the entire secret key after roughly $|sk|$ signing queries by injecting `set_bit` faults to sk before it is loaded by the signing oracle, and the faulty secret key $\tilde{sk}$ is globally used throughout the signing operation: for example, if the most significant bit of sk is set to 0 at the very beginning of signing and its output still passed the verification, then the adversary can conclude that sk has 0 in the most significant bit with high probability. In doing so, the adversary iteratively recovers sk bit-by-bit if the fault is transient. The attack above is essentially a well-known impossibility result by Gennaro et al. [GLM+04] and such an attack can be practically achieved with ineffective faults. To overcome this issue, one would require an additional strict assumption on the upper-bound of faulty signing

queries [DFMV17], or the signing algorithm needs to have some sophisticated features like self-destruct or key-updating mechanisms, which, however, are not yet widely implemented in real-world systems and are beyond the scope of this paper.

Winning Condition of Fault Adversaries As described in Fig. 3.4, the UF-fCMNA experiment keeps track of possibly faulty messages $\hat{m}$ instead of queried messages m, and it does not regard σ^* as valid forgery if it verifies with $\hat{m}$ that $\mathcal{A}$ caused in prior queries. This may appear artificial, but we introduced this condition to rule out a trivial forgery "attack": if the experiment only keeps track of queried message m_i in i-th query, and adversaries target f_5 at m_i as hash input, they obtain a valid signature $\hat{\sigma}_i$ on message $\hat{m}_i$, yet $\hat{m}_i$ is not stored in a set of queried messages M. Hence the adversary can trivially win UF-fCMNA game by just submitting $(\hat{\sigma}_i, \hat{m}_i)$, which of course verifies. This is not an actual attack, since what $\mathcal{A}$ does there is essentially asking for a signature on $\hat{m}_i$ from the signing oracle, and hence outputting such a signature as forgery should not be considered as a meaningful threat.

Note that the OFaultHSign oracle in Fig. 3.4 stores all queried messages in the same set M, whether the adversary $\mathcal{A}$ decides to inject a fault (i.e. $\phi \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i\}$) or not (i.e. $\phi := Id$), and so a forgery (m^*, σ^*) output by $\mathcal{A}$ is *not* considered valid even if m^* was only queried to OFaultHSign to obtain a faulty invalid signature. For some signature algorithms and fault types this is required; for example with Fiat–Shamir type signatures (derived from a commitment recoverable identification [KLS18]), one can query OFaultHSign to get a signature (e, z) with a single bit-flip in z, and create a valid forgery by unflipping the bit.

Validity of Oracle Output The signature output by OFaultHSign does not need to verify, but it may need to be well-formed in some way. Typically we show with a hybrid argument that OFaultHSign can be simulated without use of the private key, in a similar way to OHSign. In order for simulated outputs of OFaultHSign to be indistinguishable from real outputs, simulated signatures must be correctly distributed. In [BDF+14, CM09], the security proof shows that the faulty signature is statistically close to a value drawn from the uniform distribution, so OFaultHSign can output a random value. For the Fiat–Shamir type signature schemes we study this is not the case, for some fault types the real output of OFaultHSign verifies with an appropriately faulted hash function, and our proofs must take care to maintain these properties when simulating OFaultHSign.

3.4 Security of Hedged Signatures Against Fault Attacks

In this section we establish the (in)security of the class of hedged Fiat–Shamir signatures schemes. We give here a short overview of the main intuition behind the results in Table 3.1: f_0 faults (on the (message, nonce) pair which is input to the hedged-extractor) cannot be tolerated since they allow the adversary to get two signatures with the same randomness. On the other hand f_1 faults (on the secret key input to the hedged-extractor) can be tolerated since they do not significantly change the distribution input to the hedged-extractor. If the adversary faults the output of the hedged extractor (using f_2), we cannot prove security in general (and we can list concrete attacks e.g., against the Schnorr signature schemes), but we can prove security for the specific case of Picnic2, since the output of the hedged-extractor is not used directly, but is given as input to a PRG – thus the small bias is "absorbed" by PRG security. We remark that, while present, this attack is much less devastating than the large randomness bias LRB attack on deterministic schemes (described

Table 3.1: Summary of results for UF-fCMNA security of the hedged Fiat–Shamir type construction, for all fault types. ✓ indicates a proof of UF-fCMNA security, and ✗ indicates an attack or counterexample.

<table>
<tr><th>Fault type</th><th colspan="2">ID is subset-revealing</th><th>ID not subset-revealing</th><th>XEdDSA</th><th>Picnic2</th></tr>
<tr><td>f_0</td><td colspan="3">✗ Lemma 3.11</td><td>✗</td><td>✗</td></tr>
<tr><td>f_1</td><td colspan="3">✓ Lemma 3.4</td><td>✓ Corollary 3.1</td><td>✓ Corollary 3.3</td></tr>
<tr><td>f_2</td><td colspan="3">✗ Lemma 3.13</td><td>✗</td><td>✓ Lemma 3.19</td></tr>
<tr><td>f_3</td><td colspan="3">✗ Lemma 3.12</td><td>✗</td><td>✗ §3.6</td></tr>
<tr><td>f_4</td><td colspan="2">✓ Lemma 3.10</td><td>✗ Lemma 3.15</td><td>✗</td><td rowspan="4">✓ Corollary 3.3</td></tr>
<tr><td>f_5</td><td colspan="3">✓ Lemma 3.7</td><td rowspan="4">✓ Corollary 3.1</td></tr>
<tr><td>f_6</td><td colspan="3">✓ Lemma 3.8</td></tr>
<tr><td>f_7</td><td colspan="2">✓ Lemma 3.9</td><td>✗ Lemma 3.14</td></tr>
<tr><td>f_8, f_9, f_{10}</td><td colspan="3">✓ Lemma 3.6</td></tr>
</table>

in Section 3.2.3). With the LRB attack, the adversary only needs two signatures to recover the full key, while the attack we will show on Schnorr signature requires a significant amount of faulty biased signatures as input in practice. This indicates that hedged constructions do, to some extent, mitigate the effect of faults on the synthetic randomness.

The hedged approach does not help when the adversary faults the input to the commitment function (via f_3), since in this case the adversary can attempt to set the bits of the secret key one at the time and check if the output signature is valid or not. Note that in some kinds of ID schemes like Schnorr (known as *input-delayed* protocols [CPS+16a]) the secret key is not used in the commitment function. Faulting the input of the commitment function can still lead to insecurity, e.g., in Schnorr the adversary can bias the randomness, which in turns leads to a total break of the signature scheme. Next, the adversary can fault the output of the commitment function (via f_4): this leads to insecurity in general, e.g., in Schnorr this also leads to randomness bias. However, for a large class of ID schemes (which we call *subset-revealing*), including Picnic2, this fault does not lead to insecurity: intuitively either the adversary faults something that will be output as part of the response (which can easily be simulated by learning a non-faulty signature and then applying the fault on the result), or it is not part of the output and therefore irrelevant. Attacking the input or the output of the random oracle used to derive the challenge (f_5 and f_6) does not lead to insecurity, since the distribution of the random oracle does not change due to the fault (note that this would not be the case for deterministic signatures, where this kind of fault would be fatal). Faults against the input of the response function (via f_7) can break non-subset revealing signatures (once again, we can show that this fault can be used to break Schnorr signatures), but do not help the adversary in the case of a subset-revealing signature like Picnic2: similar to the case of f_4 faults, we use the fact that if the response function only outputs subsets of its input, faulting part of the input either has no effect or can be efficiently simulated given a non-faulty signature. Similarly, faults against the output of the response function or the input/output of the serialization function (fault types f_8, f_9, f_{10}) can also be easily simulated from a non-faulty signature.

We expand this high-level intuition into full proofs by carefully measuring the concrete security loss in the reductions which is introduced by the different kind of faults. More precisely, we present a concrete reduction from UF-KOA to $\{f_1, f_4, \ldots, f_{10}\}$-UF-fCMNA security for schemes derived from subset-revealing ID schemes, and to $\{f_1, f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMNA when ID is non-subset-revealing. Our theorems generalize and adapt results

from [BPS16] and [KMP16] without introducing significant additional concrete security loss. Then in Section 3.4.7, we describe attacks for the remaining fault types (f_0, f_2 and f_3), completely characterizing the security of generic $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID},\mathsf{CSF}],\mathsf{HE}]$ signature schemes for fault types $f_0,\ldots,f_{10}$.

3.4.1 Main Positive Result

Theorem 3.1 ($\mathsf{UF\text{-}KOA} \to \mathsf{UF\text{-}fCMNA}$)**.** *Let* ID *be a canonical identification protocol and* CSF *be a canonical serialization function for* ID*. Suppose* ID *satisfies the same properties as in Lemma 3.2 and it is subset revealing, and moreover, let us assume that* $\mathcal{A}$ *does not query the same* (m,n) *pair to* OFaultHSign *more than once. Then if* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID},\mathsf{CSF}]$ *is* UF-KOA *secure,* $\mathsf{HFS} := \mathbf{R2H}[\mathsf{FS},\mathsf{HE}]$ *is* $\{f_1,f_4,\ldots,f_{10}\}$-UF-fCMNA *secure in the random oracle model. Concretely, given* $\{f_1,f_4,\ldots,f_{10}\}$*-adversary* $\mathcal{A}$ *against* HFS *running in time* t*, and making at most* Q_s *queries to* OFaultHSign*,* Q_h *queries to* H *and* Q_{he} *queries to* HE*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) \le 2\cdot\left(\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s+Q_h)Q_s}{2^{\alpha-1}} + Q_s\cdot\epsilon_{HVZK}\right),$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t *plus* $Q_{he}\cdot|sk|$ *invocations of* Sign *and* Verify *of* FS*. Moreover, if we do not assume the subset-revealing property of* ID *and assume all the other conditions above, then we have that* HFS *is* $\{f_1,f_5,f_6,f_8,f_9,f_{10}\}$-UF-fCMNA *secure.*

Proof. The proof is two-fold. See Lemmas 3.4 and 3.5. □

For the rest of this section we will assume that ID satisfies the properties in Lemma 3.2. As a first step, we give a reduction from UF-fCMA to UF-fCMNA security, and then we later give a reduction from UF-KOA to UF-fCMA. We observe that the UF-CMA-to-UF-CMNA reduction in Lemma 3.3 is mostly preserved, even in the presence of 1-bit faults on sk as a hedged extractor key. However, our proof shows that such a fault does affect the running time of the adversary because the reduction algorithm needs to go through all secret key candidates queried to random oracle *and* their faulty bit-flipped variants. We present a proof in the full version.

Lemma 3.4 (F-UF-fCMA $\to F\cup\{f_1\}$-UF-fCMNA)**.** *Suppose the fault adversary* $\mathcal{A}$ *does not query the same* (m,n) *pair to* OFaultHSign *more than once. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID},\mathsf{CSF}]$ *is* F-UF-fCMA *secure, then* $\mathsf{HFS} := \mathbf{R2H}[\mathsf{FS},\mathsf{HE}]$ *is* F'-UF-fCMNA *secure in the random oracle model, where* $F' = F\cup\{f_1\}$*. Concretely, given an* F'*-adversary* $\mathcal{A}$ *against* HFS *running in time* t*, and making at most* Q_s *queries to* OFaultHSign*,* Q_h *queries to* H *and* Q_{he} *queries to* HE*, one can construct* F*-adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) \le 2\cdot\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}),$$

where $\mathcal{B}$ *makes at most* Q_s *queries to its signing oracle* OFaultSign *and* Q_h *queries to its hash oracle, and has running time* $t'\approx t+Q_{he}\cdot|sk|$*.*

Remarks. Our reduction above crucially relies upon the assumption that adversaries are not allowed to query the same (m,n) pair. Without this condition, OFaultHSign must return a faulty signature derived from the same randomness ρ if the same (m,n) is queried

twice, and thus one could not simulate it using OFaultSign as an oracle, since OFaultSign uses the fresh randomness even if queried with the same message m. In fact, by allowing the same (m, n) query the hedged construction HFS degenerates to a deterministic scheme and thus the SSND or LRB type fault attacks would become possible as we saw in Section 3.2.3. For the same reason, once we allow the adversaries to mount a fault f_0 on (m, n) right before HE is invoked during the signing query, the security is completely compromised. We will revisit this issue as a negative result in Lemma 3.11.

Lemma 3.5 (UF-KOA → UF-fCMA)**.** *Suppose* ID *is subset revealing. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* $\{f_4, \ldots, f_{10}\}$-UF-fCMA *secure in the random oracle model. Concretely, given* $\{f_4, \ldots, f_{10}\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OFaultSign*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t*. If we do not assume the subset-revealing property of* ID *and assume all the other conditions above, then we have that* FS *is* $\{f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMA *secure.*

Proof. We obtain the results by putting together Lemmas 3.6 to 3.10 for FS derived from subset-revealing ID, and Lemmas 3.6 to 3.8 for FS derived from non-subset-revealing ID. The proofs for these lemmas appear in the full version.

Our proof extends the UF-KOA-to-UF-CMA reduction from Lemma 3.2. We show that UF-KOA security of a randomized Fiat–Shamir signature scheme FS can be broken by a successful UF-fCMA adversary $\mathcal{A}$ by constructing an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and simulates OFaultSign without using sk. We denote the random oracle and hash table in UF-fCMA experiment (resp. UF-KOA experiment) by H and HT (resp. H′ and HT′).

Preparation of Public Key Upon receiving pk in the UF-KOA game, $\mathcal{B}$ forwards pk to $\mathcal{A}$.

Simulation of Random Oracle Queries Upon receiving a random oracle query H(a, m, pk) from $\mathcal{A}$, $\mathcal{B}$ forwards the input (a, m, pk) to its own random oracle (H′ from the UF-KOA game) and provides $\mathcal{A}$ with the return value.

Simulation of Faulty Signing Queries Suppose $\mathcal{A}$ chooses to use a fault function f_{j_i} in each faulty signing oracle query $i \in [Q_s]$. Then $\mathcal{B}$ answers i-th query by simulating the signature on m_i (or $\hat{m}_i$ if $\mathcal{A}$ chooses to apply f_5 to the message as hash input) using only pk as described in the lemma for f_{j_i}. Notice that the simulations are independent except they share the random oracle H and the set M storing (possibly faulty) queried messages. The hash input $(\hat{a}_i, \hat{m}_i, \hat{pk})$ in each signature simulation has at least $(\alpha - 1)$ bits of min-entropy (see the simulation in Lemma 3.7). Because HT has at most $Q_h + Q_s$ existing entries, $\mathcal{B}$ fails to program the random oracle with probability at most $(Q_h + Q_s)/2^{\alpha-1}$ for each query. Moreover, $\mathcal{A}$ distinguishes the simulated signature from the one returned by the real signing oracle OFaultHSign with probability at most ϵ_{HVZK} for each query, since we use the special c/s/p-HVZK simulator $\mathcal{M}$ to derive a signature in every simulation.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in UF-fCMA game with probability at

most

$$\frac{(Q_h + Q_s)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK}.$$

Forgery Suppose that at the end of the experiment $\mathcal{A}$ outputs its forgery (m^*, σ^*) that verifies and $m^* \notin M = \{\hat{m}_i : i \in [Q_s]\}$. (Recall from Fig. 3.4 that M stores possibly faulty messages $\hat{m}_i$ here instead of queried messages m_i, and thus $\mathcal{A}$ cannot win the game by simply submitting a signature on some faulty message that has been used for random oracle programming.) This means that the reconstructed transcript $(a^*, e^*, z^*) \leftarrow \mathsf{CDF}(\sigma^*, pk)$ satisfies

$$\mathsf{V}(a^*, e^*, z^*, pk) = 1 \quad \text{and} \quad \mathsf{H}(a^*, m^*, pk) = e^*.$$

Here we can guarantee that the $\mathrm{HT}[a^*, m^*, pk]$ has not been programmed by signing oracle simulation since m^* is fresh, i.e., $m^* \notin M$. Hence we ensure that $e^* = \mathrm{HT}[a^*, m^*, pk]$ has been directly set by $\mathcal{A}$, and $e^* = \mathrm{HT}'[a^*, m^*, pk]$ holds due to the hash query simulation. This implies (m^*, σ^*) is a valid forgery in the UF-KOA game as well.

□

3.4.2 Faulting Serialization Input/Output and Response Output

As a warm-up, we begin with the simplest analysis where faults do not have any meaningful impact on the signing oracle simulation. As we will show below, faulting with f_8, f_9 and f_{10} has no more security loss than the plain UF-KOA-to-UF-CMA reduction [KMP16] does.

Lemma 3.6 (UF-KOA $\rightarrow$ $\{f_8, f_9, f_{10}\}$-UF-fCMA (informal))**.** *If* FS $:=$ **FS**[ID, CSF] *is* UF-KOA *secure, then* FS *is* $\{f_8, f_9, f_{10}\}$-UF-fCMA *secure in the random oracle model.*

Remark As we briefly remarked after Definition 3.5, Lemma 3.6 holds for any instantiation of serialization as long as CSF and CDF are efficiently computable.

3.4.3 Faulting Challenge Hash Input

Recall that f_5 is the fault type that allows the attacker to fault the input (a, m, pk) to the hash function used to compute the challenge. Here we prove that randomized Fiat–Shamir signature schemes are secure against this type of fault attack, under the same conditions required for the plain UF-KOA-to-UF-CMA reduction [KMP16]. Note that the proof of lemma below introduces a slight additional security loss compared to the plain UF-KOA-to-UF-CMA reduction because `set_bit` faults to the hash input increase the failure probability of random oracle programming.

Lemma 3.7 (UF-KOA $\rightarrow$ $\{f_5\}$-UF-fCMA (informal))**.** *If* FS $:=$ **FS**[ID, CSF] *is* UF-KOA *secure, then* FS *is* $\{f_5\}$-UF-fCMA *secure in the random oracle model.*

3.4.4 Faulting Challenge Hash Output

Recall that f_6 is the fault type that allows the attacker to fault the challenge hash function output, i.e., he can fault the bit string $e = \mathsf{H}(a, m, pk)$. We show that, unlike the fault with f_5, this type of fault does not introduce any additional loss in concrete security as long as the Resp function fails for invalid challenges outside the challenge space D_H.

Lemma 3.8 (UF-KOA $\to$ $\{f_6\}$-UF-fCMA)**.** *If* FS $:=$ **FS**[ID, CSF] *is* UF-KOA *secure, then* FS *is* $\{f_6\}$-UF-fCMA *secure in the random oracle model.*

Remarks The above lemma relies on the fact that faulty $\tilde{e}_i$ is necessarily a "well-formed" challenge. For example, the challenge in some subset-revealing schemes has a specific structure (e.g., a list of pairs (c_i, p_i) where the c_i are distinct, as in Picnic2). Computing Resp with a malformed challenge may cause σ to leak private information. This is why we required Definition 3.1 to have the condition that Resp validates $\tilde{e}_i \in D_h$ and otherwise returns $\perp$. This way, the signing algorithm does not leak information when a malformed challenge is input to the response phase, and eventually outputs $\perp$ as a signature because CSF is sound with respect to invalid response (see Definition 3.4).

Note that the proof can be generalized to the multi-bit fault setting. More specifically, the random oracle programming becomes unnecessary for output replacement faults (i.e. f_6 applies `set_bit` to every bit of e) because in that case the fault adversary would no longer be able to observe any relation between faulty $\tilde{e}_i$ and the original, unfaulty e.

3.4.5 Faulting Response Input

Next we prove the security against tampering function f_7, which lets an attacker fault the input (sk, e, St) to the Resp function. We only guarantee security assuming that the signature scheme is based on a subset revealing identification protocol (see Definition 3.3), and Resp and CSF make sure to rule out invalid challenge and response, respectively. As we will see in the next section, Picnic2 satisfies these additional properties.

Lemma 3.9 (UF-KOA $\to$ $\{f_7\}$-UF-fCMA)**.** *Suppose* ID *is subset revealing. If* FS $:=$ **FS**[ID, CSF] *is* UF-KOA *secure, then* FS *is* $\{f_7\}$-UF-fCMA *secure in the random oracle model.*

Remark Intuitively, subset revealing ID schemes are secure against faults on St because the adversary only obtains what they could have computed by changing non-faulty signatures by themselves. On the other hand, the Schnorr signature scheme is not secure against tampering with f_7 and we describe concrete fault attacks in Lemma 3.14.

As we remarked after Definition 3.3, one can consider a highly inefficient version of Schnorr signature that enumerates all possible responses in St and opens one of them. In doing so, the Resp function avoids any algebraic operations involving sk and ρ, and we can mitigate the risk of faulty response input attacks described above. This countermeasure is of course impractical since the challenge space is too large, but it illustrates a concrete case where subset revealing ID schemes are more robust against fault attacks, in our model.

3.4.6 Faulting Commitment Output

Recall that a fault of type f_4 allows the attacker to fault the output of $\mathsf{Com}(sk; \rho)$, the commitment function in the first step of the ID scheme. Here we prove that randomized Fiat–Shamir signature schemes are secure against this type of fault attack, under the same conditions as ones in Lemma 3.9.

Lemma 3.10 (UF-KOA $\to$ $\{f_4\}$-UF-fCMA)**.** *Suppose* ID *is subset revealing. If* FS $:=$ **FS**[ID, CSF] *is* UF-KOA *secure, then* FS *is* $\{f_4\}$-UF-fCMA *secure in the random oracle model.*

3.4.7 Negative Results

Here we show that fault attacks of type f_0, f_2 and f_3 are not mitigated by the hedged construction for an ID scheme with the same properties as in Theorem 3.1.

Lemma 3.11. *There exist canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_0\}$-UF-fCMNA *secure.*

Proof. We consider the Schnorr scheme that returns (e, z) as a signature , for which $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ is known to be UF-CMA secure and therefore $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ is UF-CMNA secure due to Lemma 3.3. Our $\{f_0\}$-adversary's strategy is as follows. The adversary first calls OFaultHSign with some (m, n) without fault (i.e. $\phi = Id$) to obtain a legitimate signature (e, z). Next, the adversary calls OFaultHSign with $\phi = \texttt{flip\_bit}_i$, $j = 0$ and (m', n), where m' is identical to m except at the i-th bit. This way, it can fault m' back to m before the invocation of HE and hence the signature is derived from the same ρ as in the previous query, while the challenge and response are different since $e' = \mathsf{H}(a, m', pk)$ and $z = \rho + e' \cdot sk \mod q$. Hence we can recover sk with the SSND attack in Section 3.2.3 and break the scheme. □

Lemma 3.12. *There exist canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_3\}$-UF-fCMNA *secure.*

Proof. We describe a simple attack that works for the Picnic ID scheme. Recall that f_3 is applied to input of $\mathsf{Com}(sk; \rho)$. When querying OFaultHSign, the attacker uses `set_bit` to set the i-th bit of sk, denoted sk_i to 0, then observes whether the signature output is valid. If so, then the true value of sk_i is 0, and if not, then sk_i is one. By repeating this for each of the secret key bits, the entire key may be recovered. Some ID schemes may include internal checks and abort if some computations are detected to be incorrect relative to the public key, in this case the attacker checks whether OFaultHSign aborts. □

Note that Lemma 3.12 only applies to ID schemes where sk is used by the Com function. For the Schnorr scheme and other so-called *input delayed protocols* [CPS+16a], sk is only used by the Resp function. In this way subset-revealing ID schemes and input delayed ID schemes have the opposite behavior, since subset-revealing schemes do not use sk in the Resp function, but they must use it in the Com function.

The sensitivity of ephemeral randomness ρ in Schnorr-like schemes is well known, and once the attacker obtains sufficiently many biased signatures, the secret key can be recovered by solving the so-called *hidden number problem (HNP)* [BV96]. Previous works have shown that even a single-bit bias helps to recover sk by making use Bleichenbacher's solution to HNP [Ble00, AFG+14]. However, the currently known algorithms for the HNP do not give an asymptotically efficient attack, they only reduce the concrete security of the scheme sufficiently to allow a practical attack on some parameter sets. For instance, with the current state-of-the-art algorithm based on Bleichenbacher's attack found in the literature [TTA18a, Theorem 2], one can practically break 1-bit biased signatures instantiated over 192-bit prime order groups, using $2^{29.6}$ signatures as input, and with $2^{29.6}$ space and $2^{59.2}$ time, which is tractable for computationally well-equipped adversaries as of today.

To attack Schnorr-like schemes with f_3, the adversary would instead target the randomness ρ to cause a single-bit bias in it, and this situation is essentially same as faulting

with f_2. Such an attack would be also powerful enough to recover the entire signing key, which we describe below.

Lemma 3.13. *Relative to an oracle for the hidden number problem, there exist a non-subset revealing canonical* ID *scheme such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but neither* $\{f_2\}$-UF-fCMNA *nor* $\{f_3\}$-UF-fCMNA *secure.*

Proof. We describe an attack that works for the Schnorr signature scheme. Recall that both f_2 and f_3 can tamper with ρ in Schnorr, as its St contains the randomness ρ. If f_2 or f_3 is `set_bit` and always targets at the most significant bit of ρ to fix its value, the attacker can introduce 1-bit bias in ρ.

Relative to an oracle for the HNP, the Schnorr scheme with unbiased ρ remains secure, however, the scheme with biased ρ is broken. We must assume here that the HNP oracle does not help an attacker break the Schnorr scheme with unbiased nonces (otherwise the Theorem is trivial). It is easy to see that the HNP with uniformly random nonces does not give a unique solution – the adversary is given a system of Q_s equations with $Q_s + 1$ unknowns, so a direct application of the HNP oracle does not help. However, there may be other ways to use the HNP oracle, so we must make the assumption. □

For fault types f_7 and f_4, we have shown that $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ is secure assuming ID is subset-revealing. The following two lemmas give counterexamples when ID is not subset revealing, showing that canonical ID schemes are not generically secure for faults f_7 and f_4.

Lemma 3.14. *There exist non-subset-revealing canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_7\}$-UF-fCMNA *secure.*

Proof. We describe two attacks that work for the Schnorr signature scheme.

- If f_7 is `set_bit` and targeted at sk, the adversary can use the strategy of Lemma 3.12 to learn each bit of sk by checking whether the faulty signatures pass verification.
- If f_7 is `flip_bit` and targeted at the most significant bit of $St = \rho$, the adversary obtains (e, z') such that $z' = e \cdot sk + f_7(\rho)$, and he can recover the "faulty" commitment $a' = [f_7(\rho)]G$. Recall that the non-faulty commitment $a = [\rho]G$ satisfies $\mathsf{H}(a, m, pk) = e$, so the adversary can learn 1-bit of ρ by checking whether $\mathsf{H}(a' + [2^{\ell_\rho - 1}]G, m, pk) = e$ or $\mathsf{H}(a' - [2^{\ell_\rho - 1}]G, m, pk) = e$ holds, where ℓ_ρ is the bit length of ρ. Since we now have the most significant bit of ρ, we use the same argument as in Lemma 3.13 to show the scheme is vulnerable to fault attacks.

□

Lemma 3.15. *There exist non-subset-revealing canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_4\}$-UF-fCMNA *secure.*

Proof. Recall that f_4 is applied to (a, St), the output of Com. In the Schnorr signature scheme, St contains the per-signature ephemeral value ρ, which is the output of the hedged extractor. Therefore, the same attack as described in Lemma 3.14 for f_7-faults can be mounted with an f_4-fault. □

3.5 Analysis of XEdDSA

In this section we apply the results of Section 3.4 to the XEdDSA signature scheme. The scheme is presented in the full version [AOTZ19]. The associated ID scheme is the Schnorr ID scheme (denoted ID-Schnorr). Then we define $\mathsf{Schnorr} := \mathbf{FS}[\mathsf{ID\text{-}Schnorr}, \mathsf{CSF}]$ and $\mathsf{XEdDSA} := \mathbf{R2H}[\mathsf{Schnorr}, \mathsf{HE}]$, where CSF returns (a, z). We start by establishing some well-known properties of ID-Schnorr. Proof is given in the full version [AOTZ19]. As noted in Section 3.2 ID-Schnorr is not subset-revealing.

Lemma 3.16. ID-Schnorr *is perfect HVZK (therefore* $\epsilon_{HVZK} = 0$*) and has* 2λ *bits of min-entropy.*

UF-KOA Security Let $\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Schnorr}}(\mathcal{A})$ be the (concrete) UF-KOA security of Schnorr against an adversary $\mathcal{A}$ running in time t. As non-hedged XEdDSA is identical to Schnorr in the UF-KOA setting, the concrete analysis for Schnorr of [KMP16, Lemmas 3.5-3.7] and [PS00a, Lemma 8] are applicable. We do not repeat those results here (as they are lengthy and don't add much to the present paper), but instead state our results in terms of $\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Schnorr}}(\mathcal{A})$. We can now apply the results of Section 3.4.

Corollary 3.1. *XEdDSA is* $\{f_1, f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMNA *secure.*

Proof. We've shown above that ID-Schnorr is perfect HVZK (so $\epsilon_{HVZK} = 0$) and has $\alpha = 2\lambda$ bits of min-entropy. Then we can apply Theorem 3.1, to obtain

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{XEdDSA}}(\mathcal{A}) \leq 2\left(\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Schnorr}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{2\lambda-1}}\right)$$

□

Remaining fault types. We now consider the faults of type f_0, f_2, f_3, f_4, and f_7 where we can't prove security. For each of these, we have given an attack elsewhere in the paper, for Schnorr signatures, but that also applies to XEdDSA. For type f_0 see Lemma 3.11, for types f_2 and f_3 see Lemma 3.13, for type f_4 see Lemma 3.15 and for type f_7 see Lemma 3.14.

3.6 Analysis of Picnic2

In this section we analyze the Picnic2 variant of the Picnic signature scheme using our formal model for fault attacks. Since Picnic is constructed from a subset-revealing ID scheme, more of the results from Section 3.4 apply, reducing our effort in this section. We use ID-Picnic2 to denote the ID scheme, and $\mathsf{Picnic2} := \mathbf{FS}[\mathsf{ID\text{-}Picnic2}, \mathsf{CSF}]$ and $\mathsf{HS\text{-}Picnic2} := \mathbf{R2H}[\mathsf{Picnic2}, \mathsf{HE}]$ to denote the randomized and hedged signature schemes. Proofs for this section, and details of the signature scheme are in the full version [AOTZ19]. We begin with some general properties of Picnic2.

ID-Picnic2 is a subset-revealing ID scheme. Note that its St consists of $\{h_j, h'_j, \mathsf{sd}^*_j, \{\hat{z}_{j,\alpha}\}, \mathsf{st}_{j,i}, \mathsf{com}_{j,i}, \mathsf{msgs}_{j,i}\}_{j\in[M],i\in[n]}$ and Resp simply reveals a subset of it depending on a challenge $\mathcal{C}$ and $\mathcal{P}$.

The Picnic2 specification is an instance of R2H. The specification recommends a hedging construction that is an instance of the **R2H** construction from Section 3.3. In this

case, the salt and random seeds are derived deterministically from $sk\|m\|pk\|n$ where n is a 2κ-bit random value (acting as the nonce in the notation of Section 3.3). The function HE is instantiated with the SHA-3 based derivation function SHAKE. The security analysis in [ZCD+19] applies to the randomized version of the signature scheme, so we must use Lemma 3.3 to establish UF-CMNA security of the hedged variant.

Lemma 3.17. *For security parameter κ, $\mathsf{ID\text{-}Picnic2}$ has $\alpha \geq 2\kappa + 256$ bits of min-entropy.*

The next corollary shows that $\mathsf{Picnic2}$ is secure against key-only attacks, and it follows from the unforgeability security proof of $\mathsf{Picnic2}$ from [ZCD+19].

Corollary 3.2. *The signature scheme $\mathsf{Picnic2}$ is $\mathsf{UF\text{-}KOA}$ secure, when the hash functions H_0, H_1, H_2 and G are modeled as random oracles with 2κ-bit outputs, and key generation function Gen is (t, ϵ_{OW})-one-way.*

In particular, we have that

$$\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}}(\mathcal{A}) \leq \frac{3{Q_h}^2}{2^{2\lambda}} + 2\epsilon_{OW} + \frac{Q_h}{2^{\lambda}} .$$

Lemma 3.18. $\mathsf{ID\text{-}Picnic2}$ *is a special c-HVZK proof, under the following assumptions: the hash functions H_0, H_1 and H_2 are modeled as random oracles, key generation is a (t, ϵ_{OWF})-secure one-way function, and the PRG is (t, ϵ_{PRG})-secure. Simulated transcripts are computationally indistinguishable from real transcripts, and all polynomial-time distinguishing algorithms succeed with probability at most*

$$\epsilon_{HVZK} \leq Q_s \left((n+2)\tau \cdot \epsilon_{PRG} + \frac{M}{2^{2\lambda}} + \epsilon_{OWF} \right) .$$

where q_0 and q_2 are the number of queries to H_0 and H_2, Q_s is the number of transcripts that are simulated, λ is the security parameter, and (M, n, τ) are parameters of the scheme.

We can now apply our results from Section 3.4.

Corollary 3.3. $\mathsf{HS\text{-}Picnic2}_r$ *is $\{f_1, f_4, \ldots, f_{10}\}$-$\mathsf{UF\text{-}fCMNA}$ secure.*

Proof. Recall that by Corollary 3.2, $\mathsf{Picnic2}_r$ is $\mathsf{UF\text{-}KOA}$ secure with

$$\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}_r}(\mathcal{A}) \leq \frac{3{Q_h}^2}{2^{2\lambda}} + 2\epsilon_{OW} + \frac{Q_h}{2^{\lambda}}$$

and the min-entropy α is $2\kappa + 256$ as shown in Lemma 3.17.

We can apply Theorem 3.1, to obtain

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HS\text{-}Picnic2}}(\mathcal{A}) \leq \frac{6{Q_h}^2}{2^{2\lambda}} + 4\epsilon_{OW} + \frac{2Q_h}{2^{\lambda}} + \frac{(Q_s + Q_h)Q_s}{2^{2\lambda+254}} + 2Q_s \cdot \epsilon_{HVZK},$$

where ϵ_{HVZK} is given in Lemma 3.18.

□

Fault type f_2 Recall that f_2 is a fault on ρ, the output of the hedged extractor. Intuitively, $\mathsf{HS\text{-}Picnic2}$ is $\{f_2\}$-$\mathsf{UF\text{-}fCMNA}$ secure since ρ is not used directly, ρ is the list of sd^*_j values, which are used as input to a PRG when deriving the $\mathsf{sd}_{i,j}$ values. Applying a 1-bit fault to a sd^*_j value reduces the min-entropy by at most one bit, so only a small change to the security proof and analysis is required. Concretely we have:

Lemma 3.19. HS-Picnic2 *is* $\{f_2\}$-UF-fCMNA *secure.* $\mathbf{Adv}^{\text{UF-fCMNA}}_{\text{HS-Picnic2}}(\mathcal{A})$ *is the same as given in Corollary 3.3, except that* α *is reduced by 1.*

Fault type f_3 Recall that f_3 faults are applied to $\mathsf{com}(f_3(sk;\rho))$. By setting bits of sk, the attacker can recover sk with an IFA.

3.7 Concluding Remarks

This paper explored the effects of bit-tampering fault attacks on various internal values in hedged Fiat–Shamir signing operations, within the provable security methodology. Our security model is general enough to capture a large class of signatures, but also fine-grained enough to cover existing attacks surveyed in Section 3.2.3. We remark, however, that there are several more advanced, yet practically relevant fault types that are not covered by our model: 1) faulting global parameters, 2) multiple bit and word faults, 3) faults within the com and Resp functions, 4) multiple faults per signature query, and 5) persisting faults. A detailed discussion for each is given in the full version [AOTZ19], to illustrate the limitations of our analysis. Each of these issues makes an interesting direction for future work.

Part II

New Constructions

Chapter 4

Two-Round Multi-Party Signing from Lattices

4.1 Introduction

In recent years, distributed signing protocols have been actively researched, motivated by many new applications for instance in the blockchain domain. One of the main motivations to construct a distributed signature is reducing the risk of compromising the secret key, which could occur in various ways, for instance as a result of attacks on cryptographic devices. In this paper, we study two similar classes of distributed signing protocols that can be constructed from standard lattice-based computational hardness assumptions, namely *n-out-of-n distributed signature* and *multi-signature* schemes.

n-out-of-n signature. An n-out-of-n signature is a special case of general t-out-of-n threshold signature. At a high level, the parties involved in n-out-of-n signature first invoke a key generation protocol in a way that each individual party P_j learns a share sk_j of the single signing key sk, but sk is unknown to anyone, and then they interact with each other to sign the message of interest. The required security property can be informally stated as follows: If all n parties agree to sign the message then they always produce a single signature that can be verified with a single public key pk; if at most $n-1$ parties are corrupted, it is not possible for them to generate a valid signature. Several recent works have studied threshold versions of ECDSA, arguably the most widely deployed signature scheme, both in the 2-out-of-2 variant [Lin17a, DKLs18, CCL+19] and in the more general t-out-of-n case [GGN16, GG18, LN18, DKLs19, DOK+20, CGG+20, CCL+20, DJN+20, GKSS20].

However it is well known that ECDSA does not withstand quantum attacks since it is based on discrete log, and it is therefore important to study post-quantum alternatives which support threshold signing. Despite this, very few works have considered n-out-of-n (or t-out-of-n) lattice-based signatures. Bendlin et al. [BKP13] proposed a threshold protocol to generate Gentry–Peikert–Vaikuntanathan signature [GPV08]; Boneh et al. [BGG+18] developed a universal thresholdizer that turns any signature scheme into a non-interactive threshold one, at the cost of using relatively heavy threshold fully homomorphic encryption.

Fiat–Shamir with aborts. Neither of the above previous papers investigated signatures following the *Fiat–Shamir with Aborts (FSwA)* paradigm due to Lyubashevsky [Lyu09, Lyu12], which was specifically designed for lattice-based signatures and is nowadays one of the most efficient and popular approaches to constructing such schemes. Recall that in

standard Fiat-Shamir signatures, the scheme is based on an underlying 3-move Σ-protocol where transcripts are of form (w, c, z) and where c is a random challenge. This interactive protocol is then turned into a non-interactive signature scheme by choosing the challenge as the hash of the first message w and the message to be signed. The FSwA paradigm follows the same approach, with the important difference that the signer (prover) is allowed to abort the protocol after seeing the challenge. Only the non-aborting instances are used for signatures, and it turns out that this allows the signer to reduce the size of the randomness used and hence reduces signature size. This comes at the cost of marginally larger signing time because some (usually quite small) fraction of the protocol executions are lost due to aborts.

Examples of single-user schemes based on FSwA include Dilithium [LDK+19] and qTESLA [BAA+19], which are round-3 and round-2 candidates of the NIST Post-Quantum Cryptography Standardization process. Cozzo and Smart [CS19] recently estimated the concrete communication and computational costs required to build a distributed version of these schemes by computing Dilithium and qTESLA with generic multi-party computation. They pointed out inherent performance issues with MPC due to the mixture of both linear and non-linear operations within the FSwA framework.

Given all this, an obvious open question is to construct secure n-out-of-n protocols specifically tailored to the FSwA, while also achieving small round complexity.

Multi-signature. A multi-signature protocol somewhat resembles n-out-of-n signature and allows a group of n parties holding a signing key $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ to collaboratively sign the same message to obtain a single signature. However, multi-signature protocols differ from n-out-of-n signing in the following ways: (1) there is no dedicated key generation protocol, and instead each party P_j locally generates its own key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ and publish pk_j before signing (so-called the *plain public-key model* [BN06]), (2) the group of signing parties is usually not fixed, and each party can initiate the signing protocol with a dynamically chosen set of parties associated with $L = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, and (3) the verification algorithm usually doesn't take a single fixed public key, and instead takes a particular set of public keys L that involved in the signing protocol. Hence, roughly speaking, multi-signatures have more flexibility than n-out-of-n signatures in terms of the choice of co-signers, at the cost of larger joint public key size and verification time (unless more advanced feature like key aggregation [MPSW19] is supported).

Schnorr vs FSwA. There is a long line of research that starts from Schnorr's signature scheme [Sch90] and follows the standard Fiat–Shamir paradigm to build distributed signatures [SS01, NKDM03, AF04, GJKR07, KG20] and multi-signatures [MOR01, BN06, BCJ08, MWLD10, STV+16, MPSW19, NRSW20, NRS21, AB21, BD21]. In particular, Drijvers et al. [DEF+19] recently discovered a flaw of the existing *two-round* Schnorr-based multi-signatures, with a novel concurrent attack relying on the generalized birthday algorithm of Wagner [Wag02]. They accordingly proposed mBCJ scheme, a provably secure variant of Bagherzhandi et al.'s BCJ scheme [BCJ08].

Unlike distributed n-out-of-n signatures, several *three* or *four-round* multi-signatures based on FSwA are already present in the literature. Bansarkhani and Sturm [ES16] extended Güneysu–Lyubashevsky–Pöppelmann (GLP) [GLP12] signature and proposed the first multi-signature following the FSwA paradigm, which was recently followed by multiple similar variants [MJ19, TLT19, TE19, FH19, FH20]. Relying on the syntactic similarities between Schnorr and FSwA-style signatures, these protocols essentially borrow the ideas of Schnorr-based counterparts; for instance, [ES16] can be considered as a

direct adaptation of Bellare and Neven's three-round Schnorr-like multi-signature [BN06]. However, as explained below, the security proofs of all these protocols are either incomplete or relying on a non-standard hardness assumption, where the underlying problem only emerges in the Fiat–Shamir *with aborts* setting. Therefore, we are also motivated to construct a provably secure multi-signature protocol within this paradigm, while making the most of useful observations from the discrete log setting.

Issue with "aborts". We first observe that there is an inherent issue when constructing distributed FSwA signatures. Just like earlier constructions in the discrete log setting [BN06, NKDM03] previous FSwA multi-signatures ask all parties to start doing what is essentially a single-user FSwA signature, and always reveal the first "commit" message of the underlying Σ-protocol. Then all these messages are added up and the sum is hashed, together with the message to be signed, in order to obtain the challenge. This means that all executions are revealed, whether they abort or not. An important issue with the FSwA approach is that, currently there is no known general way to prove the underlying Σ-protocol zero-knowledge in case of aborts [BCK+14, §3.2],[ESS+19, §4],[BBE+18, BBE+19],[Lyu19, p.26]. As a result, the signer should not reveal any of the aborted executions since otherwise the scheme cannot be proved secure. This issue is not serious in a single-user scheme, since the Σ-protocol is made non-interactive in the random oracle model anyway and there is no reason why the signer would reveal aborted executions.

In an interactive setting, the standard approach to circumvent the issue is to send a commitment to the first Σ-protocol message and only reveal it if the rejection sampling is successful. However, the previous FSwA multi-signatures skipped this subtle step. Thus the concurrent work by Fukumitsu and Hasegawa [FH20] (who constructed a FSwA-style multi-signature proven secure in QROM) had to rely on an additional non-standard assumption (which they call "rejected" LWE), while publicly available security proofs of other similar constructions [ES16, FH19, TLT19, MJ19, TE19] do not explain how to simulate the aborted executions. Despite the lack of such discussion in the proofs there are no known concrete attacks against the existing schemes, and it may be that one could patch the problem by making additional non-standard assumptions, or by carefully choosing the parameter such that the additional assumptions hold unconditionally. Still, it is paramount to strive to find protocols which can be proven secure relying on well-established computational hardness assumptions like LWE and SIS.

4.1.1 Contributions

FSwA-based distributed signatures with full security proof. In this paper we construct FSwA-type n-out-of-n distributed and multi-signature protocols solely relying on the hardness of learning with errors (LWE) and short integer solution (SIS) problems. Our constructions can be seen as distributed variants of the fast Dilithium-G signature scheme [DLL+18]. As a first step, we circumvent the aborts issue mentioned above by utilizing Baum et al.'s *additively homomorphic* commitment scheme [BDL+18], which is currently the most efficient construction based on lattices and relies on the hardness of Module-LWE and Module-SIS problems. This results in a provably secure (in the classical random oracle model), three-round n-out-of-n signature protocol DS_3 [DOTT20][1].

[1]It is still an open question whether the aborts issue can instead be resolved by careful parameter choice, allowing to simulate the rejected transcripts without any additional assumptions. But we are aware of on-going work in this direction. If the question is answered in the affirmative our three-round protocol

First two-round protocols. Previous FSwA-based multi-signatures required at least three rounds of interaction. On the other hand, as most recent discrete log-based solutions indicate [DEF+19, KG20, NRSW20, NRS21, AB21], two rounds is a natural goal because this clearly seems to be minimal for a distributed signature protocol based on the Fiat-Shamir paradigm: we first need to determine what should be hashed in order to get the challenge in the underlying Σ-protocol. This must (for security) include randomness from several players and hence requires at least one round of interaction. After this we need to determine the prover's answer in the Σ-protocol. This cannot be computed until the challenge is known and must (for security) require contributions from several players, and we therefore need at least one more round.

In this paper, we show that the application of homomorphic commitment not only resolves the issue with aborts, but also makes it possible to reduce the round complexity to two rounds. We do this by adding a *trapdoor* feature to the commitment scheme (a separate contribution that we discuss in more detail below). This results in a two-round, n-out-of-n signature protocol DS_2 presented in Section 4.3. With a slight modification this n-out-of-n protocol can be also turned into a two-round multi-signature scheme in the plain public key model. We describe a multi-signature variant MS_2 in Section 4.4.

Our main two-round result highlights several important similarities and differences which emerge when translating a discrete log-based protocol to lattice-based one. The approaches taken in our two-round protocols are highly inspired by mBCJ discrete log-based multi-signature by Drijvers et al. [DEF+19] In particular, we observe that it is crucial for two-round protocols to use message-dependent commitment keys (as in mBCJ) instead of a single fixed key for all signing attempts (as in the original BCJ [BCJ08]), because otherwise the proof doesn't go through. Drijvers et al. only presented a full security proof for the protocol in the *key verification model*, in which each co-signer has to submit a zero-knowledge proof of knowledge of the secret key. Our protocols confirm that a similar approach securely transfers to the lattice setting even under different security models: distributed n-out-of-n signature with dedicated key generation phase, and multi-signature in the plain public key model.

Lattice-based trapdoor commitment. We turn Baum et al.'s scheme into a trapdoor commitment in Section 4.5, so that the two-round protocols DS_2 and MS_2 are indeed instantiable with only lattice-based assumptions. We make use of the lattice trapdoor by Micciancio and Peikert [MP12] to generate a trapdoor commitment key in the ring setting. The only modification required is that the committer now samples randomness from the discrete Gaussian distribution instead of the uniform distribution. This way, the committer holding a trapdoor of the commitment key can equivocate a commitment to an arbitrary message by sampling a small randomness vector from the Gaussian distribution. Such randomness is indeed indistinguishable from the actual randomness used in the committing algorithm. Since only a limited number of lattice-based trapdoor commitment schemes are known [GSW13, CHKP10, DM14, GVW15, LNTW19] our technique may be of independent interest.

---

could be proven secure even without a commitment. However, the use of homomorphic commitment is crucial for constructing our new two-round protocols, which is our main contribution.

4.1.2 Technical Overview

Our protocols are based on Dilithium signature scheme, which works over rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$ defined with an appropriate irreducible polynomial $f(X)$ (see preliminaries for more formal details). Here we go over the core ideas of our construction by considering simple 2-out-of-2 signing protocols. The protocols below can be generalized to an n-party setting in a straightforward manner. We assume that each party P_j for $j = 1, 2$ owns a secret signing key share $\mathbf{s}_j \in R^{\ell+k}$ which has small coefficients, and a public random matrix $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$. The joint public verification key is defined as $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$, where $\mathbf{t} = \bar{\mathbf{A}}(\mathbf{s}_1 + \mathbf{s}_2) \bmod q$. In the actual protocols pk also needs to be generated in a distributed way, but here we omit the key generation phase for brevity's sake.

Naïve approach. We first present a naïve (insecure) way to construct a 2-party signing protocol from FSwA. If the reader is familiar with CoSi Schnorr multi-signature [STV+16] this construction is essentially its lattice-based, 2-out-of-2 variant. In this protocol the parties P_j for $j = 1, 2$ involved in signing the message μ work as follows.

1. P_j samples a randomness $\mathbf{y}_j$ from some distribution $D^{\ell+k}$ defined over $R^{\ell+k}$ (which is typically the uniform distribution over a small range or discrete Gaussian), and then sends out the first message of FSwA $\mathbf{w}_j = \bar{\mathbf{A}}\mathbf{y}_j \bmod q$.
2. P_j locally derives a joint challenge $c = \mathsf{H}(\mathbf{w}_1 + \mathbf{w}_2, \mu, \mathsf{pk})$ and performs the rejection sampling $\mathsf{RejSamp}(c\mathbf{s}_j, \mathbf{z}_j)$ with $\mathbf{z}_j = c\mathbf{s}_j + \mathbf{y}_j$; if the result of $\mathsf{RejSamp}(c\mathbf{s}_j, \mathbf{z}_j)$ is "reject" then P_j sets $\mathbf{z}_j \coloneqq \perp$. After exchanging $\mathbf{z}_j$'s if $\mathbf{z}_1 = \perp$ or $\mathbf{z}_2 = \perp$ (i.e., either of the parties aborts), then the protocol restarts from the step 1.
3. Each party outputs $(\mathbf{w}, \mathbf{z}) \coloneqq (\mathbf{w}_1 + \mathbf{w}_2, \mathbf{z}_1 + \mathbf{z}_2)$ as a signature on μ.

Note that the rejection sampling step is needed to make the distribution of $\mathbf{z}_j$ independent of a secret $\mathbf{s}_j$. The verification algorithm checks that the norm of $\mathbf{z}$ is small, and that $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} = \mathbf{w} \pmod q$ holds, where the challenge is recomputed as $c \leftarrow \mathsf{H}(\mathbf{w}, \mu, \mathsf{pk})$. One can easily check that the signature generated as above satisfies correctness, thanks to the linearity of the SIS function $f_{\bar{\mathbf{A}}}(\mathbf{x}) = \bar{\mathbf{A}}\mathbf{x} \bmod q$. However, we observe that an attempt to give a security proof fails due to two problems. Suppose the first party $\widetilde{P}_1$ is corrupt and let us try to simulate the values returned by honest P_2, whenever queried by the adversary.

First, since the protocol reveals $\mathbf{w}_2$ whether P_2 aborts or not, the joint distribution of rejected transcript $(\mathbf{w}_2, c, \perp)$ has to be simulated. As mentioned earlier, there is no known way to simulate it; in fact, the honest verifier zero knowledge (HVZK) of FSwA is only proven for "non-aborting" cases in the original work by Lyubashevsky [Lyu09, Lyu12, Lyu19] and its successors. Note that the obvious fix where players hash the initial messages and only reveal them if there is no abort will not work here: the protocols need to *add* the initial messages together before obtaining the challenge c in order to reduce signature size, only the sum is included in the signature. So with this approach the initial messages must be known in the clear before the challenge can be generated.

The second problem is more generic and could also occur in the standard Fiat–Shamir-style two party signing: if P_2 sends out $\mathbf{w}_2$ first, then the simulator does not know $\mathbf{w}_1$. In FS-style constructions, the usual strategy for signing oracle query simulation is to first sample a challenge c by itself, generate a simulated transcript $(\mathbf{w}_2, c, \mathbf{z}_2)$ by invoking a special HVZK simulator on c, and then program the random oracle H such that its output is fixed to a predefined challenge c. In the two-party setting, however, derivation of the

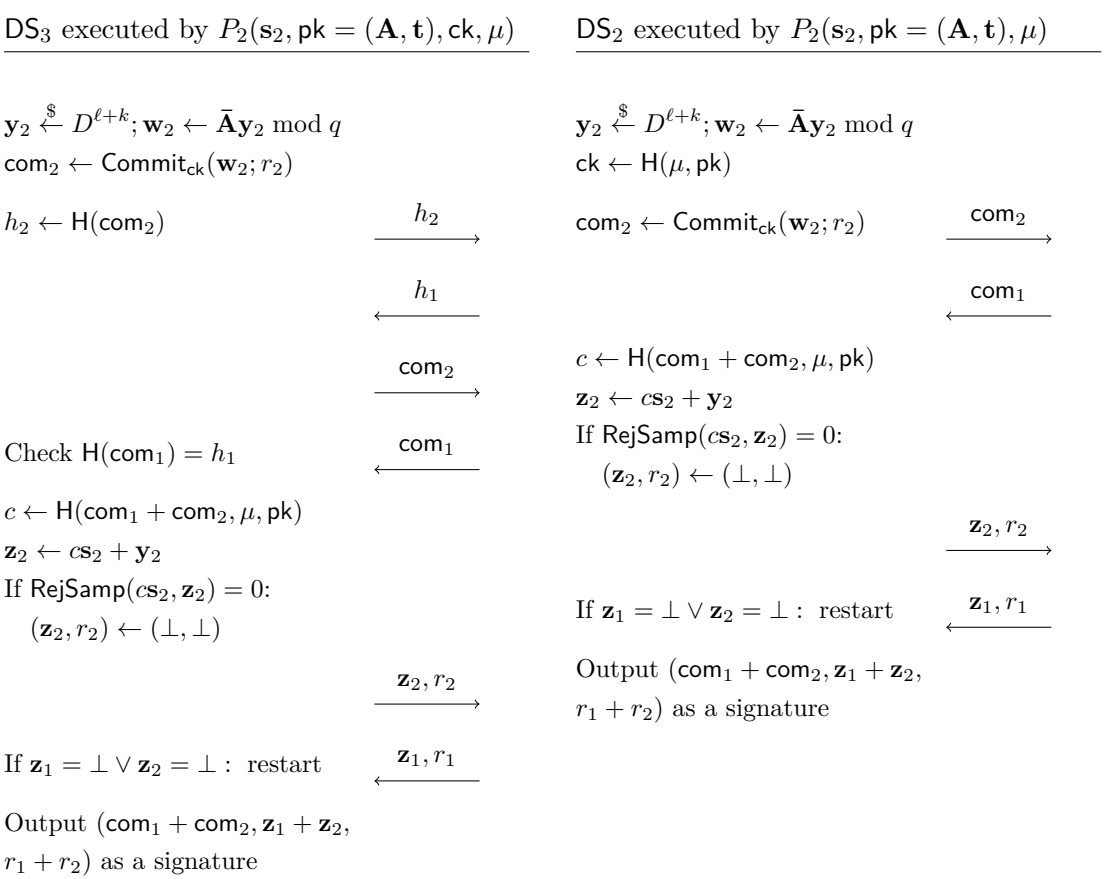

Figure 4.1: Comparison of different instantiations of FSwA-based 2-party signing protocols

joint challenge $c = \mathsf{H}(\mathbf{w}_1 + \mathbf{w}_2, \mu, \mathsf{pk})$ requires contribution from $\widetilde{P}_1$ and thus there is no way for the simulator to program H in advance. Not only the proof doesn't go through, but also this naïve construction is amenable to a concrete attack, which allows malicious $\widetilde{P}_1$ to create a valid forgery by adaptively choosing $\mathbf{w}_1$ after seeing $\mathbf{w}_2$. In [DOTT20] we describe this attack relying on a variant of Wagner's generalized birthday problem [Wag02, HJ10].

Homomorphic commitment to simulate aborts. We now present a provably secure, intermediate protocol that circumvents the above issues. See DS_3 in Fig. 4.1. To address the first issue with aborts, each player P_j now commits to an initial Σ-protocol message $\mathbf{w}_j$ using an *additively homomorphic* commitment com_j. Thanks to the hiding property, each party leaks no useful information about $\mathbf{w}_j$ until the rejection sampling is successful, and thus it is now possible to simulate a rejected transcript $(\mathsf{com}_j, c, \perp)$. Then P_j broadcasts a hash based commitment to com_j, to deal with the second issue. Once all parties have done this, com_j's are revealed in the next round and checked against the hashes. Once the com_j's are known, they can be added together in a meaningful way by the homomorphic property, and we then hash the sum and the message to get the challenge. The verification now receives a signature consisting of three elements $(\mathsf{com}, \mathbf{z}, r)$ and simply checks that $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} \pmod q$ and r form a correct opening to com, where the challenge is recomputed as $c \leftarrow \mathsf{H}(\mathsf{com}, \mu, \mathsf{pk})$.

We note that the extra round for hash commitment is a standard technique, previously used in multiple three-round protocols, such as Nicolosi et al. [NKDM03] and Bellare

and Neven [BN06] in the discrete log setting, and Bansarkhani and Sturm [ES16] in their FSwA-based instantiation. This way, the simulator for honest P_2 can successfully extract corrupt $\widetilde{P}_1$'s share com_1 by keeping track of incoming queries to H (when modeled as a random oracle), and program H such that $\mathsf{H}(\mathsf{com}_1 + \mathsf{com}_2, \mu, \mathsf{pk}) := c$ before revealing com_2. In [DOTT20] we provide a formal security proof for DS_3 by showing a reduction to Module-LWE *without* relying on the forking lemma [PS00a, BN06].[2] This is made possible by instantiating the construction with unconditionally binding commitment, which allows us to avoid rewinding the adversary and apply the *lossy identification* technique by Abdalla et al. [AFLT16].

One efficiency issue is, that the protocol has to be restarted until *all* parties pass the rejection sampling step simultaneously. All previous FSwA-based multi-signatures also had the same issues, but we can mitigate by running sufficiently many parallel executions of the protocol at once, or by carefully choosing the parameters for rejection sampling. To further reduce the number of aborts, we chose to instantiate the protocol with Dilithium-"G" [DLL+18] instead of the one submitted to NIST competition [LDK+19].

Trapdoor commitment to avoid the extra round. Although DS_3 is secure, the first round of interaction may seem redundant, since the parties are essentially "committing to a commitment". We show that the extra hash commitment round can be indeed dropped by adding a trapdoor feature to the commitment scheme, which allows the so-called *straight-line simulation* technique due to Damgård [Dam00]. We present our main two-round protocol DS_2 in Fig. 4.1. This way, the simulation of honest P_2 does not require the knowledge of corrupt $\widetilde{P}_1$'s commitment share; instead, the simulator can now simply send a commitment com_2 (to some random value) and then later equivocate to an arbitrary value using the known trapdoor td associated with a trapdoored commitment key tck. Concretely, the simulator need not program the random oracle this time, and instead derives a challenge $c \leftarrow \mathsf{H}(\mathsf{com}_1 + \mathsf{com}_2, \mu, \mathsf{pk})$ as the real honest party would do. Now the simulator invokes a (special) HVZK simulator with c as input, to obtain a transcript $(\mathbf{w}_2, c, \mathbf{z}_2)$. With some constant probability it equivocates com_2 to $\mathbf{w}_2$, or otherwise sends out $\perp$ to simulate aborts. We also stress that the *per-message commitment key* $\mathsf{ck} \leftarrow \mathsf{H}(\mu, \mathsf{pk})$ is crucial in the two-round protocol; if a single ck is used across all signing attempts, then a concurrent attack similar to the one against the naïve construction becomes applicable (see [DOTT20]). Unlike the three-round protocol, we present a security proof relying on the forking lemma and we reduce the security to both Module-SIS and Module-LWE assumptions; since a trapdoor commitment can at most be computationally binding, we must extract from the adversary two different openings to the same commitment in order to be able to reduce security to the binding property. We leave for future work a tighter security proof, as well as a proof in the quantum random oracle model.

Two-round multi-signature. We can now convert to a two-round multi-signature scheme in the plain public key model: following Bellare–Neven [BN06] the protocol now generates *per-user challenges* $c_j = \mathsf{H}(\mathbf{t}_j, \sum_j \mathsf{com}_j, \mu, L)$ for each user's public key $\mathbf{t}_j \in L$, instead of interactively generating the fixed joint public key $\mathbf{t}$ in advance. The verification algorithm is adjusted accordingly: given $(\mathsf{com}, \mathbf{z}, r)$ and a list of public keys L, the verifier checks that $\bar{\mathbf{A}}\mathbf{z} - \sum_j c_j \mathbf{t}_j \pmod q$ and r form a correct opening to com, where c_j's are recomputed as in the signing protocol. Section 4.4 formally describes our MS_2 protocol with security proof.

[2] We include this for completeness since, while the three-round protocol itself is not novel, to the best of our knowledge there has been no publicly available complete security proof solely relying on Module-LWE.

	Functionality	# Rounds	Type	Assumption	Building blocks
[BGG+18]	t-out-of-n	1	FSwA	Lyubashevsky'12	Threshold FHE
[BKP13]	t-out-of-n	1	H&S	GPV'08	Honest-majority MPC
Our DS_3	n-out-of-n	3	FSwA	MLWE	Homomorphic COM
Our DS_2	n-out-of-n	2	FSwA	MLWE & MSIS	Homomorphic TDCOM
[FH20]	Multisig	3	FSwA	MLWE & rMLWE / QROM	—
Our MS_2	Multisig	2	FSwA	MLWE & MSIS	Homomorphic TDCOM

Table 4.1: Comparison with previous lattice-based multi-party signing protocols with (publicly available) full security proofs. "FSwA" denotes Fiat–Shamir with aborts signatures of [Lyu09, Lyu12] and "H&S" denotes hash-and-sign-type signature of [GPV08], respectively.

4.1.3 Related Work

The FSwA paradigm was first proposed by Lyubashevsky [Lyu09, Lyu12] and many efficient signature schemes following this framework have been devised, such as GLP [GLP12], BLISS [DDLL13], Dilithium [LDK+19] and qTESLA [BAA+19]. Bansarkhani and Sturm [ES16] extended GLP signature and proposed the first multi-signature following the FSwA paradigm. Since then several variants appeared in the literature: four-round protocol with public key aggregation [MJ19], three-round protocol with tight security proof [FH19] and proof in QROM [FH20], ID-based blind multi-signature [TLT19] and ID-based proxy multi-signature [TE19]. However, as mentioned earlier the security proofs for all these multi-signatures are either incomplete or rely on a non-standard heuristic assumption. Choi and Kim [CK16] proposed a linearly homomorphic multi-signature from lattices trapdoors. Kansal and Dutta [KD20] constructed a single-round multi-signature scheme relying on the hardness of SIS, which was soon after broken by Liu et al. [LTT20]. Several lattice-based threshold *ring signatures* exist in the literature, such as Cayrel et al. [CLRS10], Bettaieb and Schrek [BS13], and Torres et al. [TSSK20]. Döroz et al. [DHSS20] and Boudgoust and Adeline Roux-Langlois [BR21] devised lattice-based *aggregate signature schemes* relying on rejection sampling. Very recently, Esgin et al. [EEE20] developed FSwA-based *adaptor signatures* with application to blockchains.

Our two-round protocols rely on trapdoor commitment to enable the straight-line simulation of ZK. The trick is originated in a concurrent ZK proof by Damgård [Dam00] and similar ideas have been extensively used in the ZK literature [BKLP15, CPS+16a, COSV17b, COSV17a], to turn honest verifier ZK proof into full-fledged ZK. Moreover, recent efficient lattice-based ZK proofs [dLS18, ESS+19, YAZ+19, BLS19, ESLL19] also make use of Baum et al.'s additively homomorphic commitment. The issue of revealing the first "commit" message in the FSwA framework has been also discussed by Barthe et al. [BBE+18] in the context of masking countermeasure against side-channel attacks, and they used Baum et al.'s commitment to circumvent the issue. The homomorphic lattice-based trapdoor commitment could also be instantiated with GSW-FHE [GSW13], homomorphic trapdoor functions [GVW15], Chameleon hash [CHKP10, DM14] or mercurial commitment [LNTW19].

Comparison with Fukumitsu and Hasegawa [FH20] A concurrent work due to Fukumitsu and Hasegawa proposed a multi-signature scheme based on Dilithium. Our MS_2 and DS_3 have different pros and cons compared with their construction. First, although both MS_2 and [FH20] are proven secure in the plain public-key model, our scheme requires only two rounds of interaction while theirs is a three-round protocol that closely

follows the existing paradigm of [BN06]. Due to the "abort" issue we raised earlier, their security proof required an additional hardness assumption *rejected* Module-LWE (rMLWE). However, our proof for MS_2 relies on the forking lemma incurring a quadratic security loss, while [FH20] circumvents that to give a proof in the QROM. In fact, both DS_3 and [FH20] essentially benefit from the same lossy identification techniques of [AFLT16, KLS18] to avoid rewinding and to obtain tighter security proofs. It is therefore an interesting follow-up work to prove security of DS_3 or [FH20] patched with the committed first-round messages in the QROM assuming only Module-LWE. Another advantage of [FH20] is that their construction considers signature size compression techniques present in the original Dilithium signature scheme and it is thus closer to the scheme submitted to the NIST PQC competition.

Comparison with Bendlin et al. [BKP13] An entirely different approach to constructing threshold signatures based on lattices relies not on the Fiat–Shamir with aborts paradigm, but on GPV hash-and-sign signatures [GPV08]. This approach was introduced by Bendlin et al. in [BKP13], who described how to implement Peikert's hash-and-sign signatures [Pei10] in a multiparty setting. Compared to the approach in this paper, it has the advantage of realizing the same distributed signature scheme (e.g., with the same size bound for verification) independently of the number of parties, and in particular, signature size does not grow with the number of parties. Moreover, it supports more general access structure than the full threshold considered in this paper (although their protocol does not withstand dishonest majority for the sake of information-theoretic security, while our protocol does tolerate up to $n-1$ corrupt parties). Its main downside, however, is that the most expensive part of Peikert's signing algorithm, namely the offline lattice Gaussian sampling phase, is carried out using *generic multiparty computation* (this is the first step of the protocol π_{Perturb} described in [BKP13, Fig. 23]). This makes it difficult to estimate the concrete efficiency of Bendlin et al.'s protocol, but since the Peikert signature scheme is fairly costly even in a single-user setting, the protocol is unlikely to be practical.

In contrast, while our protocols do use discrete Gaussian sampling, it is only carried out *locally by each party*, and it is Gaussian sampling over $\mathbb{Z}$ rather than a lattice, which is considerably less costly. Furthermore, while we also use lattice trapdoors as a proof technique in the trapdoor commitment scheme of our two-round protocol, trapdoor Gaussian sampling is never carried out in the actual protocol, only in the simulation (the actual protocol has no trapdoor). Thus, our protocols entirely avoid the expensive machinery present in Bendlin et al.'s scheme, and have a fully concrete instantiation (at the cost of signatures increasing in size with the number of parties).

Comparison with Boneh et al. [BGG+18] A previous work of Boneh et al. [BGG+18] proposed a general solution to non-interactive, t-out-of-n threshold signatures based on the *universal thresholdizer* enabled by FHE. Their approach roughly works as follows: each party is supplied with a common FHE-encrypted signing key sk and one share of the Shamir-secret-shared FHE decryption key. Upon receiving the message μ to be signed, they evaluate a circuit computing $\mathsf{Sign}(\mathsf{sk}, \mu)$ on the public ciphertext as input, and output a partial decryption of the FHE-encrypted signature. These partial evaluations of the Sign function can then be combined into a valid signature using the reconstruction algorithm of the Shamir secret sharing. Similar to [BKP13], the approach of [BGG+18] has the advantage of making the signature size as well as the verification bound independent of the number of participants. As the authors remark in §7, however, their construction comes with a few caveats. Since each party must homomorphically compute a challenge

hash function and rejection sampling if instantiated with a FSwA-based Sign function, the actual running time of sign operations may be slow in practice. Moreover, the universal thresholdizer assumes a *trusted setup*, where a trusted authority provides each user in the system with a share of the FHE decryption key. In contrast, our protocol describes a simple, concrete multi-party key generation phase and evaluations of hash functions and rejection sampling are carried out in the clear by each party.

4.2 Preliminaries

Notations. For positive integers a and b such that $a < b$ we use the integer interval notation $[a, b]$ to denote $\{a, a+1, \ldots, b\}$; we use $[b]$ as shorthand for $[1, b]$. If S is a set we write $s \xleftarrow{\$} S$ to indicate sampling s from the uniform distribution defined over S; if D is a probability distribution we write $s \xleftarrow{\$} D$ to indicate sampling s from the distribution D; if we are explicit about the set S over which the distribution D is defined then we write $D(S)$; if $\mathcal{A}$ is an algorithm we write $s \leftarrow \mathcal{A}$ to indicate assigning an output from $\mathcal{A}$ to s.

4.2.1 Polynomial Rings and Discrete Gaussian Distribution

In this paper most operations work over rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$, where q is a modulus, N is a power of two defining the degree of $f(X)$, and $f(X) = X^N + 1$ is the $2N$-th cyclotomic polynomial. Following [DLL+18], we consider *centered modular reduction* $\bmod^{\pm} q$: for any $v \in \mathbb{Z}_q$, $v' = v \bmod^{\pm} q$ is defined to be a unique integer in the range $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ such that $v' = v \bmod q$. We define the norm of $v \in \mathbb{Z}_q$ such that $\|v\| \coloneqq |v \bmod^{\pm} q|$. Now we define the L^p-norm for a (vector of) ring element $\mathbf{v} = (\sum_{i=0}^{N-1} v_{i,1} X^i, \ldots, \sum_{i=0}^{N-1} v_{i,m} X^i)^T \in R^m$ as follows:

$$\|\mathbf{v}\|_p \coloneqq \left\| (v_{0,1}, \ldots, v_{N-1,1}, \ldots, v_{0,m}, \ldots, v_{N-1,m})^T \right\|_p .$$

We rely on the following *key set* $S_\eta \subseteq R$ parameterized by $\eta \geq 0$ consisting of small polynomials:

$$S_\eta = \{x \in R : \|x\|_\infty \leq \eta\} .$$

Moreover the *challenge set* $C \subseteq R$ parameterized by $\kappa \geq 0$ consists of small and sparse polynomials, which will be used as the image of random oracle H_0:

$$C = \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\} . \tag{4.1}$$

The discrete Gaussian distribution over R^m is defined as follows.

Definition 4.1 (Discrete Gaussian Distribution over R^m)**.** *For $\mathbf{x} \in R^m$, let $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{v}\|_2^2 / s^2)$ be a* Gaussian function *of parameters $\mathbf{v} \in R^m$ and $s \in \mathbb{R}$. The* discrete Gaussian distribution $D^m_{\mathbf{v},s}$ *centered at $\mathbf{v}$ is*

$$D^m_{\mathbf{v},s}(\mathbf{x}) \coloneqq \rho_{\mathbf{v},s}(\mathbf{x}) / \rho_{\mathbf{v},s}(R^m)$$

where $\rho_{\mathbf{v},s}(R^m) = \sum_{\mathbf{x} \in R^m} \rho_{\mathbf{v},s}(\mathbf{x})$.

In what follows we omit the subscript $\mathbf{v}$ if $\mathbf{v} = \mathbf{0}$ and write D^m_s as a shorthand. When s exceeds the so-called *smoothing parameter* $\eta(R^m) \leq \omega(\sqrt{\log(mN)})$ of the ambient

space, then the discrete Gaussians $D_{R^m-\mathbf{v},s} = D^m_{\mathbf{v},s} - \mathbf{v}$ supported on all cosets of R^m are statistically close, and hence D^m_s behaves qualitatively like a continuous Gaussian of standard deviation $\sigma = s/\sqrt{2\pi}$. The condition on s will be satisfied for all the discrete Gaussians in this paper, and hence σ will be called the standard deviation (even though it technically holds only up to negligible error). For the same reason, we will always be in a setting where the following fact [MP13, Theorem 3.3][ESLL19, Lemma 9] holds.

Lemma 4.1 (Sum of Discrete Gaussian Samples)**.** *Suppose s exceeds the smoothing parameter by a factor $\geq \sqrt{2}$. Let $\mathbf{x}_i$ for $i \in [n]$ be independent samples from the distribution D^m_s. Then the distribution of $\mathbf{x} = \sum_i \mathbf{x}_i$ is statistically close to $D^m_{s\sqrt{n}}$.*

4.2.2 Lattice Problems

Below we define two standard lattice problems over rings: module short integer solution (MSIS) and learning with errors (MLWE). We also call them MSIS/MLWE *assumption* if for any probabilistic polynomial-time adversaries the probability that they can solve a given problem is negligible. Note that the latter k elements of $\mathbf{s}$ correspond to the error term of MLWE.

Definition 4.2 ($\mathsf{MSIS}_{q,k,\ell,\beta}$ problem)**.** *Given a random matrix $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$ find a vector $\mathbf{x} \in R_q^{\ell+k}$ such that $[\mathbf{A}|\mathbf{I}] \cdot \mathbf{x} = \mathbf{0}$ and $0 < \|\mathbf{x}\|_2 \leq \beta$.*

Definition 4.3 ($\mathsf{MLWE}_{q,k,\ell,\eta}$ problem)**.** *Given a pair $(\mathbf{A}, \mathbf{t}) \in R_q^{k\times\ell} \times R_q^k$ decide whether it was generated uniformly at random from $R_q^{k\times\ell} \times R_q^k$, or it was generated in a way that $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}, \mathbf{s} \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$ and $\mathbf{t} \coloneqq [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s}$.*

4.2.3 Fiat–Shamir with Aborts Framework and Dilithium-G

Algorithm 6 Key generation

Require: $\mathsf{pp} = (R_q, k, \ell, \eta, B, s, M)$
Output: $(\mathsf{sk}, \mathsf{pk})$

1: $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$
2: $\bar{\mathbf{A}} \coloneqq [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$
3: $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k; \mathbf{s} \coloneqq \begin{bmatrix}\mathbf{s}_1\\ \mathbf{s}_2\end{bmatrix}$
4: $\mathbf{t} \coloneqq \bar{\mathbf{A}}\mathbf{s}$
5: $\mathsf{sk} \coloneqq \mathbf{s}$
6: $\mathsf{pk} \coloneqq (\bar{\mathbf{A}}, \mathbf{t})$
7: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 7 Signature verification

Require: $\mathsf{pk}, (\mathbf{z}, c), \mu, \mathsf{pp}$
1: If $\|\mathbf{z}\|_2 \leq B$ and $c = \mathsf{H}_0(\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}, \mu, \mathsf{pk})$:
2: **return** 1
3: Otherwise: **return** 0

Algorithm 8 Signature generation

Require: $\mathsf{sk}, \mathsf{pk}, \mu, \mathsf{pp} = (R_q, k, \ell, \eta, B, s, M)$
Output: valid signature pair $(\mathbf{z}, c)$

1: $(\mathbf{y}_1, \mathbf{y}_2) \xleftarrow{\$} D_s^\ell \times D_s^k; \mathbf{y} \coloneqq \begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}$
2: $\mathbf{w} \coloneqq \bar{\mathbf{A}}\mathbf{y}$
3: $c \leftarrow \mathsf{H}_0(\mathbf{w}, \mu, \mathsf{pk})$
4: $\mathbf{z} \coloneqq c\mathbf{s} + \mathbf{y}$
5: With prob. $\min\left(1, D_s^{\ell+k}(\mathbf{z})/(M \cdot D_{c\mathbf{s},s}^{\ell+k}(\mathbf{z}))\right)$:
6: **return** $(\mathbf{z}, c)$
7: Restart otherwise

Algorithm 9 $\mathsf{Trans}(\mathsf{sk}, c)$

$\mathbf{y} \xleftarrow{\$} D_s^{\ell+k}$
$\mathbf{w} := \bar{\mathbf{A}}\mathbf{y}$
$\mathbf{z} := c\mathbf{s} + \mathbf{y}$
With prob. $\min\left(1, D_s^{\ell+k}(\mathbf{z})/(M \cdot D_{c\mathbf{s},s}^{\ell+k}(\mathbf{z}))\right)$:
    return $(\mathbf{w}, c, \mathbf{z})$
Otherwise:
    return $(\perp, c, \perp)$

Algorithm 10 $\mathsf{SimTrans}(\mathsf{pk}, c)$

$\mathbf{z} \xleftarrow{\$} D_s^{\ell+k}$
$\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$
With prob. $1/M$:
    return $(\mathbf{w}, c, \mathbf{z})$
Otherwise:
    return $(\perp, c, \perp)$

We present a non-optimized version of Dilithium-G signature scheme in Algorithms 6 to 8, on which we base our distributed signing protocols. The random oracle is defined as $\mathsf{H}_0 : \{0,1\}^* \to C$. Due to [Lyu12, Lemma 4.4] we restate below, the maximum L^2-norm of the signature $\mathbf{z} \in R^{\ell+k}$ is set to $B = \gamma\sigma\sqrt{(\ell+k)N}$, where the parameter $\gamma > 1$ is chosen such that the probability $\gamma^{(\ell+k)N} e^{(\ell+k)N(1-\gamma^2)/2}$ is negligible.

Lemma 4.2. *For any* $\gamma > 1$, $\Pr[\|\mathbf{z}\|_2 > \gamma\sigma\sqrt{mN} : \mathbf{z} \xleftarrow{\$} D_s^m] < \gamma^{mN} e^{mN(1-\gamma^2)/2}$.

The following claim by Lyubashevsky (adapted from [Lyu12, Lemma 4.5]) is crucial for the signing oracle of FSwA to be simulatable, and also to decide the standard deviation σ as well as the expected number of repetitions M. For instance, setting $\alpha = 11$ and $t = 12$ leads to $M \approx 3$. Although M is asymptotically superconstant, t increases very slowly in practice, and hence M behaves essentially like a constant for practical security parameters (in the literature, it is often taken as 12 to ensure $\epsilon < 2^{-100}$, thereby ensuring > 100 bits of security).

Lemma 4.3. *For* $V \subseteq R^m$ *let* $T = \max_{\mathbf{v}\in V} \|\mathbf{v}\|_2$. *Fix some* t *such that* $t = \omega(\sqrt{\log(mN)})$ *and* $t = o(\log(mN))$. *If* $\sigma = \alpha T$ *for any positive* α, *then*

$$\Pr[M \geq D_s^m(\mathbf{z})/D_{\mathbf{v},s}^m(\mathbf{z}) : \mathbf{z} \xleftarrow{\$} D_s^m(\mathbf{z})] \geq 1 - \epsilon$$

where $M = e^{t/\alpha + 1/(2\alpha^2)}$ *and* $\epsilon = 2e^{-t^2/2}$.

We now present a supporting lemma which is required for Dilithium-G to be UF-CMA secure. This is almost a direct consequence of Lemma 4.3 and a similar result appears in [KLS18, Lemma 4.3] to prove the security of Dilithium signature instantiated with the uniform distribution. We remark that the simulator in Algorithm 10 can only simulate transcripts of non-abort executions in the underlying *interactive* Σ-protocol; in fact, if Trans output $\mathbf{w}$ in case of rejection as it's done in the interactive protocol then there is no known method to simulate the *joint distribution* of $(\mathbf{w}, c, \perp)$ [BCK+14, Lyu19] (without assuming some ad-hoc assumptions like rejection-DCK [BBE+18] or rejected-LWE [FH20]).

Lemma 4.4 (Non-abort Special Honest Verifier Zero Knowledge)**.** *Let* $m = \ell + k$ *and* $T = \max_{c\in C, \mathbf{s}\in S_\eta^m} \|c \cdot \mathbf{s}\|_2$. *Fix some* t *such that* $t = \omega(\sqrt{\log(mN)})$ *and* $t = o(\log(mN))$. *If* $\sigma = \alpha T$ *for any positive* α, *then for any* $c \in C$ *(as defined in (4.1)) and* $\mathbf{s} \in S_\eta^m$, *the output distribution of* $\mathsf{Trans}(\mathsf{sk}, c)$ *(Algorithm 9) is within statistical distance* ϵ/M *of the output distribution of* $\mathsf{SimTrans}(\mathsf{pk}, c)$ *(Algorithm 10), where* $M = e^{t/\alpha + 1/(2\alpha^2)}$ *and* $\epsilon = 2e^{-t^2/2}$. *Moreover,* $1/M \geq \Pr[\mathsf{Trans}(\mathbf{s}, c) \neq (\perp, c, \perp)] \geq (1-\epsilon)/M$.

Proof. The proof closely follows the rejection sampling lemma due to Lyubashevsky [Lyu12, Lemma 4.7], but since we are interested in showing *special* HVZK, we account for fixed challenge $c \in C$ given to the simulator as input. For each $c \in C$ and $\mathbf{s} \in S_\eta^m$, we define $\mathbf{v} = c \cdot \mathbf{s}$ and $S_\mathbf{v} = \left\{\mathbf{z} \in R^m : M \geq D_s^m(\mathbf{z}) / D_{\mathbf{v},s}^m(\mathbf{z})\right\}$. We first consider simplified algorithms $\mathsf{Trans}'(\mathbf{s}, c)$ and $\mathsf{SimTrans}'(c)$, which do not take the public key as input and only output $(c, \mathbf{z})$ or $(c, \perp)$. Then we have

$$\begin{aligned}
\Pr[\mathsf{Trans}'(\mathbf{s}, c) \neq (c, \perp)] &= \sum_{\mathbf{z} \in R^m} D_{\mathbf{v},s}^m(\mathbf{z}) \cdot \min\left(1, \frac{D_s^m(\mathbf{z})}{M \cdot D_{\mathbf{v},s}^m(\mathbf{z})}\right) \\
&= \sum_{\mathbf{z} \in S_\mathbf{v}} \frac{D_s^m(\mathbf{z})}{M} + \sum_{\mathbf{z} \notin S_\mathbf{v}} D_{\mathbf{v},s}^m(\mathbf{z}) \\
&\geq \sum_{\mathbf{z} \in S_\mathbf{v}} \frac{D_s^m(\mathbf{z})}{M} \geq \frac{1-\epsilon}{M}
\end{aligned}$$

where the last inequality holds from Lemma 4.3. Clearly, the upper bound is

$$\Pr[\mathsf{Trans}'(\mathbf{s}, c) \neq (c, \perp)] \leq \sum_{\mathbf{z} \in R^m} \frac{D_s^m(\mathbf{z})}{M} \leq \frac{1}{M}.$$

As $\mathbf{w}$ is not involved in the rejection sampling we also have that $1/M \geq \Pr[\mathsf{Trans}(\mathbf{s}, c) \neq (\perp, c, \perp)] \geq (1-\epsilon)/M$. Now we find the statistical distance:

$$\begin{aligned}
\Delta(\mathsf{Trans}'(\mathbf{s}, c), \mathsf{SimTrans}'(c)) =& \frac{1}{2}\Bigg(\sum_{\mathbf{z} \in R^m} \left|\Pr[\mathsf{Trans}'(\mathbf{s}, c) = (c, \mathbf{z})] - \Pr[\mathsf{SimTrans}'(c) = (c, \mathbf{z})]\right| \\
&+ \left|\Pr[\mathsf{Trans}'(\mathbf{s}, c) = (c, \perp)] - \Pr[\mathsf{SimTrans}'(c) = (c, \perp)]\right| \Bigg) \\
\leq& \frac{1}{2}\Bigg(\sum_{\mathbf{z} \in R^m} \left|\Pr[\mathsf{Trans}'(\mathbf{s}, c) = (c, \mathbf{z})] - \Pr[\mathsf{SimTrans}'(c) = (c, \mathbf{z})]\right| \\
&+ \left|\left(1 - \frac{1-\epsilon}{M}\right) - \left(1 - \frac{1}{M}\right)\right| \Bigg) \\
=& \frac{1}{2}\left(\sum_{\mathbf{z} \in R^m} \left| D_{\mathbf{v},s}^m(\mathbf{z}) \cdot \min\left(1, \frac{D_s^m(\mathbf{z})}{M \cdot D_{\mathbf{v},s}^m(\mathbf{z})}\right) - \frac{D_s^m(\mathbf{z})}{M} \right| + \frac{\epsilon}{M} \right) \\
=& \frac{1}{2}\left(\sum_{\mathbf{z} \in S_\mathbf{v}} \left| \frac{D_s^m(\mathbf{z})}{M} - \frac{D_s^m(\mathbf{z})}{M} \right| + \sum_{\mathbf{z} \notin S_\mathbf{v}} \left| D_{\mathbf{v},s}^m(\mathbf{z}) - \frac{D_s^m(\mathbf{z})}{M} \right| + \frac{\epsilon}{M} \right) \\
\leq& \frac{1}{2}\left(\sum_{\mathbf{z} \notin S_\mathbf{v}} \frac{D_s^m(\mathbf{z})}{M} + \frac{\epsilon}{M} \right) \leq \frac{1}{2}\left(\frac{\epsilon}{M} + \frac{\epsilon}{M} \right) = \frac{\epsilon}{M}
\end{aligned}$$

where the last inequality holds from Lemma 4.3. Finally, outputting $\mathbf{w}$ in non-abort cases doesn't increase the statistical distance: since the underlying identification protocol is *commitment recoverable* [KLS18], $\mathbf{w}$ can be reconstructed given c, $\mathbf{z}$ and pk. This concludes the proof. □

4.2.4 Trapdoor Homomorphic Commitment Scheme

Below we formally define a trapdoor commitment scheme with the standard security notions. The two additional properties are required by our protocols: additive homomorphism and uniform key. The lattice-based commitments described in Section 4.5 indeed satisfy all of them. The uniform property is required since our protocols rely on commitment key derivation via random oracles mapping to a key space S_{ck}, and thus its output distribution should look like the one from CGen. Many other standard schemes like Pedersen commitment [Ped92] trivially satisfy this property. The additive homomorphism is also needed to preserve the algebraic structure of the first "commit" message of FSwA. Finally, the trapdoor feature is crucial for achieving two-round protocols.

Definition 4.4 ((Trapdoor) Commitment Scheme)**.** *A trapdoor commitment scheme* TCOM *consists of the following algorithms.*

- $\mathsf{CSetup}(1^\kappa) \to \mathsf{cpp}$*: The setup algorithm outputs a public parameter* cpp *defining sets* $S_{\mathsf{ck}}, S_m, S_r, S_{\mathsf{com}}$*, and* S_{td} *and the distribution* $D(S_r)$ *from which the randomness is sampled.*
- $\mathsf{CGen}(\mathsf{cpp}) \to \mathsf{ck}$*: The key generation algorithm that samples a commitment key from* S_{ck}*.*
- $\mathsf{Commit}_{\mathsf{ck}}(m; r) \to \mathsf{com}$*: The committing algorithm that takes a message* $m \in S_m$ *and randomness* $r \in S_r$ *as input and outputs* $\mathsf{com} \in S_{\mathsf{com}}$*. We simply write* $\mathsf{Commit}_{\mathsf{ck}}(m)$ *when it uses* r *sampled from* $D(S_r)$*.*
- $\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, m) \to b$*: The opening algorithm outputs* $b = 1$ *if the input tuple is valid, and* $b = 0$ *otherwise.*
- $\mathsf{TCGen}(\mathsf{cpp}) \to (\mathsf{tck}, \mathsf{td})$*: The trapdoor key generation algorithm that outputs* $\mathsf{tck} \in S_{\mathsf{ck}}$ *and the trapdoor* $\mathsf{td} \in S_{\mathsf{td}}$*.*
- $\mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td}) \to \mathsf{com}$*: The trapdoor committing algorithm that outputs a commitment* $\mathsf{com} \in S_{\mathsf{com}}$*.*
- $\mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}, m) \to r$*: The equivocation algorithm that outputs randomness* $r \in S_r$*.*

A usual commitment scheme COM *is a special case of* TCOM*: it only consists of* CSetup, CGen, Commit, *and* Open.

Correctness. TCOM *(resp.* COM*) is correct if for any* $m \in S_m$

$$\Pr\left[\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, m) \to 1 : \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa); \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ r \xleftarrow{\$} D(S_r); \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m; r) \end{array}\right] = 1.$$

Hiding TCOM *(resp.* COM*) is unconditionally (resp. computationally) hiding if the following probability is negligible in* κ *for any probabilistic adversary (resp. probabilistic polynomial-time adversary)* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*.*

$$\epsilon_{\mathsf{hide}} := \left|\Pr\left[b = b' \ : \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa); \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ (m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{ck}, \mathsf{cpp}) \\ b \xleftarrow{\$} \{0,1\}; \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{com}) \end{array}\right] - \frac{1}{2}\right|$$

Binding TCOM *(resp.* COM*) is unconditionally (resp. computationally) binding if the following probability is negligible in* κ *for any probabilistic adversary (resp. probabilistic*

polynomial-time adversary) $\mathcal{A}$.

$$\epsilon_{\mathsf{bind}} \coloneqq \Pr\left[\begin{array}{ll} m \neq m' & \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa) \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, m) \rightarrow 1 \quad : & \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r', m') \rightarrow 1 & (\mathsf{com}, m, r, m', r') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right]$$

In particular, unconditionally binding implies that the following probability is also negligible in κ*, since otherwise unbounded adversaries can simply check all possible values in* S_{com}*,* S_m *and* S_r *to find a tuple that breaks binding.*

$$\epsilon_{ubind} \coloneqq \Pr\left[\begin{array}{ll} \exists(\mathsf{com}, r, m, r', m') : & \\ m \neq m' & \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa) \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, m) \rightarrow 1 \quad : & \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r', m') \rightarrow 1 & \end{array}\right]$$

Secure Trapdoor TCOM *has a secure trapdoor if for any* $m \in S_m$*, the statistical distance* ϵ_{td} *between* $(\mathsf{ck}, m, \mathsf{com}, r)$ *and* $(\mathsf{tck}, m, \mathsf{com}', r')$ *is negligible in* κ*, where*

- $\mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa); \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}); r \xleftarrow{\$} D(S_r); \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m; r)$*, and*
- $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{TCGen}(\mathsf{cpp}); \mathsf{com}' \leftarrow \mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td}); r' \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}', m)$.

Definition 4.5 (Uniform Key)**.** *A commitment key is said to be* uniform *if the output of* $\mathsf{CGen}(\mathsf{cpp})$ *follows the uniform distribution over the key space* S_{ck}.

Definition 4.6 (Additive Homomorphism)**.** *A commitment scheme is said to be* additively homomorphic *if for any* $m, m' \in S_m$

$$\Pr\left[\mathsf{Open}_{\mathsf{ck}}(\mathsf{com} + \mathsf{com}', r + r', m + m') \rightarrow 1 : \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\kappa) \\ \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ r \xleftarrow{\$} D(S_r); r' \xleftarrow{\$} D(S_r) \\ \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m; r) \\ \mathsf{com}' \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m'; r') \end{array}\right] = 1.$$

4.2.5 Security Notions for n-out-of-n Signature and Multi-Signature

We first define the n-out-of-n distributed signature protocol and its security notion. The game-based security notion below is based on the one presented by Lindell [Lin17a] for two-party signing protocol. Our definition can be regarded as its generalization to n-party setting. Following Lindell, we assume that the key generation can be invoked only once, while many signing sessions can be executed concurrently. The main difference is that, in our protocols all players have the same role, and therefore we fix wlog the index of honest party and challenger to n, who has to send out the message first in each round of interaction. This way, we assume that the adversary $\mathcal{A}$ who corrupts $P_1, \ldots, P_{n-1}$ is *rushing* by default (i.e., $\mathcal{A}$ is allowed to choose their own messages based on P_n's message).

Definition 4.7 (Distributed Signature Protocol)**.** *A distributed signature protocol* DS *consists of the following algorithms.*

- $\mathsf{Setup}(1^\kappa) \rightarrow \mathsf{pp}$*: The set up algorithm that outputs public parameters* pp *on a security parameter* κ *as input.*

$\mathsf{Exp}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A})$

1 : $Mset \leftarrow \varnothing$
2 : $pp \leftarrow \mathsf{Setup}(1^\kappa)$
3 : $(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{DS}}_n(\cdot,\cdot)}(pp)$
4 : $b \leftarrow \mathsf{Ver}(\mu^*, \sigma^*, \mathsf{pk})$
5 : **return** $(b = 1) \wedge \mu^* \notin Mset$

$\mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A})$

1 : $Mset \leftarrow \varnothing$
2 : $pp \leftarrow \mathsf{Setup}(1^\kappa)$
3 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(pp)$
4 : $(\mu^*, \sigma^*, L^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{MS}}(\cdot,\cdot)}(\mathsf{pk}, \mathsf{pp})$
5 : $b \leftarrow \mathsf{Ver}(\mu^*, \sigma^*, L^*)$
6 : **return** $(b = 1) \wedge \mathsf{pk} \in L^* \wedge (\mu^*, L^*) \notin Mset$

Figure 4.2: DS-UF-CMA and MS-UF-CMA experiments. The oracles $\mathcal{O}^{\mathsf{DS}}_n$ and $\mathcal{O}^{\mathsf{MS}}$ are described in Figs. 4.3 and 4.4. In the left (resp. right) experiment, *Mset* is the set of all inputs μ (resp. (μ, L)) such that (sid, μ) (resp. $(\mathsf{sid}, (\mu, L))$) was queried by $\mathcal{A}$ to its oracle as the first query with identifier $\mathsf{sid} \neq 0$ (resp. with any identifier sid). Note that pk in the left experiment is the public verification key output by P_n when it completes $\mathsf{Gen}_n(\mathsf{pp})$.

- $\mathsf{Gen}_j(\mathsf{pp}) \to (\mathsf{sk}_j, \mathsf{pk})$ *for every* $j \in [n]$*: The interactive key generation algorithm that is run by party* P_j*. Each* P_j *runs the protocol on public parameters* pp *as input. At the end of the protocol* P_j *obtains a secret key share* sk_j *and public key* pk*.*
- $\mathsf{Sign}_j(\mathsf{sid}, \mathsf{sk}_j, \mathsf{pk}, \mu) \to \sigma$ *for every* $j \in [n]$*: The interactive signing algorithm that is run by party* P_j*. Each* P_j *runs the protocol on session ID* sid*, its signing key share* sk_j*, public key* pk*, and message to be signed* μ *as input. We also assume that the algorithm can use any state information obtained during the key generation phase. At the end of the protocol* P_j *obtains a signature* σ *as output.*
- $\mathsf{Ver}(\sigma, \mu, \mathsf{pk}) \to b$*: The verification algorithm that takes a signature, message, and a single public key* pk *and outputs* $b = 1$ *if the signature is valid and otherwise* $b = 0$*.*

Definition 4.8 (DS-UF-CMA Security)**.** *A distributed signature protocol* DS *is said to be* DS-UF-CMA *(distributed signature unforgeability against chosen message attacks) secure, if for any probabilistic polynomial time adversary* $\mathcal{A}$*, its advantage*

$$\mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A}) \to 1\right]$$

is negligible in κ*, where* $\mathsf{Exp}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A})$ *is described in Fig. 4.2.*

Next we define the standard security notion of multi-signature protocol in the plain public-key model. The following definitions are adapted from [BN06], but the syntax is made consistent with n-out-of-n signing. The main difference from the distributed signature is, that there is no interactive key generation protocol and the adversary is not required to fix its key pair at the beginning of the game. Accordingly, the adversary can dynamically choose a set of public keys involving the challenger's key, and query the signing oracle to receive signatures. On the other hand, assuming that key aggregation is not always supported the verification algorithm takes a set of public keys, instead of a single combined public key as in the prior case. We also note that n is now the number of *maximum* number of parties involved in a single execution of signing protocol, since the size of L may vary depending on a protocol instance.

Oracle $\mathcal{O}_n^{\mathsf{DS}}(\mathsf{sid}, m)$

The oracle is initialized with public parameters pp generated by Setup algorithm. The variable flag is initially set to false.

Key Generation Upon receiving $(0, m)$, if $\mathsf{flag} = \mathsf{true}$ then return $\perp$. Otherwise do the following:

- If the oracle is queried with $\mathsf{sid} = 0$ for the first time then it initializes a machine $\mathcal{M}_0$ running the instructions of party P_n in the distributed key generation protocol $\mathsf{Gen}_n(\mathsf{pp})$. If P_n sends the first message in the key generation protocol, then this message is the oracle reply.
- If $\mathcal{M}_0$ has been already initialized then the oracle hands the machine $\mathcal{M}_0$ the next incoming message m and returns $\mathcal{M}_0$'s reply. If $\mathcal{M}_0$ concludes with local output $(\mathsf{sk}_n, \mathsf{pk})$, then set $\mathsf{flag} = \mathsf{true}$.

Signature Generation Upon receiving (sid, m) with $\mathsf{sid} \neq 0$, if $\mathsf{flag} = \mathsf{false}$ then return $\perp$. Otherwise do the following:

- If the oracle is queried with sid for the first time then parse the incoming message m as μ. It initializes a machine $\mathcal{M}_{\mathsf{sid}}$ running the instructions of party P_n in the distributed signing protocol $\mathsf{Sign}_n(\mathsf{sid}, \mathsf{sk}_n, \mathsf{pk}, \mu)$. The machine $\mathcal{M}_{\mathsf{sid}}$ is initialized with the key share and any state information stored by $\mathcal{M}_0$ at the end of the key generation phase. The message μ to be signed is included in *Mset*. If P_n sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{\mathsf{sid}}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{\mathsf{sid}}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{\mathsf{sid}}$. If $\mathcal{M}_{\mathsf{sid}}$ concludes with local output σ, then the output obtained by $\mathcal{M}_{\mathsf{sid}}$ is returned.

Figure 4.3: Honest party oracle for the distributed signing protocol.

Definition 4.9 (Multi-signature Protocol)**.** *A multisignature protocol* MS *consists of the following algorithms.*

- $\mathsf{Setup}(1^\kappa) \to \mathsf{pp}$*: The set up algorithm that outputs a public parameter* pp *on a security parameter* κ *as input.*
- $\mathsf{Gen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$*: The non-interactive key generation algorithm that outputs a key pair on a public parameter* pp *as input.*
- $\mathsf{Sign}(\mathsf{sid}, \mathsf{sk}, \mathsf{pk}, \mu, L) \to \sigma$*: The interactive signing algorithm that is run by a party* P *holding a key pair* $(\mathsf{sk}, \mathsf{pk})$*. Each* P *runs the protocol on session ID* sid*, its signing key* sk*, public key* pk*, message to be signed* μ*, and a set of co-signers' public keys* L *as input. At the end of the protocol* P *obtains a signature* σ *as output.*
- $\mathsf{Ver}(\sigma, \mu, L) \to b$*: The verification algorithm that takes a signature, message, and a set of public keys and outputs* $b = 1$ *if the signature is valid and otherwise* $b = 0$*.*

Definition 4.10 (MS-UF-CMA Security)**.** *A multisignature protocol* MS *is said to be* MS-UF-CMA *(multisignature unforgeability against chosen message attacks) secure, if for*

Oracle $\mathcal{O}^{\mathsf{MS}}(\mathsf{sid}, m)$

The oracle is initialized with public parameters pp generated by Setup algorithm.

Signature Generation Upon receiving (sid, m) do the following:

- If the oracle is queried with sid for the first time then parse the incoming message m as (μ, L). If $\mathsf{pk} \notin L$ then it returns $\perp$. Otherwise it initializes a machine $\mathcal{M}_{\mathsf{sid}}$ running the instructions of party P in the multi-signature protocol $\mathsf{Sign}(\mathsf{sid}, \mathsf{sk}, \mathsf{pk}, \mu, L)$. The machine $\mathcal{M}_{\mathsf{sid}}$ is initialized with the key pair $(\mathsf{sk}, \mathsf{pk})$ and any state information obtained during $\mathsf{Gen}(\mathsf{pp})$. The pair (μ, L) is included in *Mset*. If P sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{\mathsf{sid}}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{\mathsf{sid}}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{\mathsf{sid}}$. If $\mathcal{M}_{\mathsf{sid}}$ concludes, then the output obtained by $\mathcal{M}_{\mathsf{sid}}$ is returned.

Figure 4.4: Honest party oracle for the multi-signature protocol.

any probabilistic polynomial time adversary $\mathcal{A}$, its advantage

$$\mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A}) \to 1\right]$$

is negligible in κ, where $\mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A})$ is described in Fig. 4.2.

4.2.6 General Forking Lemma

We restate the general forking lemma from [BN06].

Lemma 4.5 (General Forking Lemma)**.** *Let Q be a number of queries and C be a set of size $|C| > 2$. Let $\mathcal{B}$ be a randomized algorithm that on input x, $h_1, \ldots, h_Q$ returns an index $i \in [0, Q]$ and a side output out. Let IGen be a randomized algorithm that we call the input generator. Let $\mathcal{F}_{\mathcal{B}}$ be a forking algorithm that works as in Fig. 4.5 given x as input and given black-box access to $\mathcal{B}$. Suppose the following probabilities.*

$$\mathrm{acc} \coloneqq \Pr[i \neq 0 : x \leftarrow \mathsf{IGen}(1^\kappa); h_1, \ldots, h_Q \xleftarrow{\$} C; (i, out) \leftarrow \mathcal{B}(x, h_1, \ldots, h_Q)]$$
$$\mathrm{frk} \coloneqq \Pr[b = 1 : x \leftarrow \mathsf{IGen}(1^\kappa); (b, out, \hat{out}) \leftarrow \mathcal{F}_{\mathcal{B}}(x)]$$

Then

$$\mathrm{frk} \geq \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{Q} - \frac{1}{|C|}\right).$$

Alternatively,

$$\mathrm{acc} \leq \frac{Q}{|C|} + \sqrt{Q \cdot \mathrm{frk}}.$$

Algorithm $\mathcal{F}_{\mathcal{B}}(x)$

Upon receiving x

1. Pick a random coin ρ for $\mathcal{B}$.
2. Generate $h_1, \ldots, h_Q \xleftarrow{\$} C$.
3. $(i, out) \leftarrow \mathcal{B}(x, h_1, \ldots, h_Q; \rho)$.
4. If $i = 0$ then return $(0, \perp, \perp)$.
5. Regenerate $\hat{h}_i, \ldots, \hat{h}_Q \xleftarrow{\$} C$.
6. $(\hat{i}, \hat{out}) \leftarrow \mathcal{B}(x, h_1, \ldots, h_{i-1}, \hat{h}_i, \ldots, \hat{h}_Q; \rho)$.
7. If $i = \hat{i}$ and $h_i \neq \hat{h}_i$ then return $(1, out, \hat{out})$
8. Else return $(0, \perp, \perp)$.

Figure 4.5: The forking algorithm $\mathcal{F}_{\mathcal{B}}$

4.3 DS_2: Two-round n-out-of-n Signing from Module-LWE and Module-SIS

4.3.1 Protocol specification and overview

This section presents our main construction: provably secure two-round n-out-of-n protocol $\mathsf{DS}_2 = (\mathsf{Setup}, (\mathsf{Gen}_j)_{j\in[n]}, (\mathsf{Sign}_j)_{j\in[n]}, \mathsf{Ver})$, formally specified in Fig. 4.7. As mentioned in Section 4.2.5 all players have the same role and hence we only present n-th player's behavior. The protocol is built on top of an additively homomorphic trapdoor commitment scheme TCOM with uniform keys (see Section 4.2.4 for the formal definitions), and we will describe concrete instances of TCOM later in Section 4.5. We go over high-level ideas for each step below.

Parameter setup. We assume that a trusted party invokes $\mathsf{DS}_2.\mathsf{Setup}(1^\kappa)$ that outputs a set of public parameters described in Table 4.2 as well as the parameter for commitment scheme cpp (obtained by internally invoking $\mathsf{TCOM}.\mathsf{CSetup}(1^\kappa)$). Most parameters commonly appear in the literature about the Fiat–Shamir with aborts paradigm (e.g. [DLL+18, Lyu12]) and we therefore omit the details here. The bit length l_1 and l_2 should be sufficiently long for the random oracle commitments to be secure. The only additional parameters are B_n and M_n, which we describe below in Section 4.3.2.

Key generation. The key generation $\mathsf{DS}_2.\mathsf{Gen}_n$ essentially follows the approach by Nicolosi et al. [NKDM03] for two-party Schnorr signing. Upon receiving public parameters, all participants first interactively generate a random matrix $\mathbf{A} \in R_q^{k\times\ell}$, a part of Dilithium-G public key. This can be securely done with simple random oracle commitments[3]; as long as there is at least one honest party sampling a matrix share correctly, the resulting combined matrix is guaranteed to follow the uniform distribution. For the exact same

[3]We remark that the "commitments" generated by H_1 and H_2 in Fig. 4.7 are not randomized, and therefore they are not hiding. In our protocol, however, all committed values have high min-entropy and this is indeed sufficient for the security proof to hold. Alternatively, one could cheaply turn them into full-fledged secure and extractable commitments by additionally hashing random strings that are to be sent out during the opening phase [Pas03].

	Parameter	Description
n		Number of parties
N		A power of two defining the degree of $f(X)$
$f(X)$	$= X^N + 1$	The $2N$-th cyclotomic polynomial
q		Prime modulus
R	$= \mathbb{Z}[X]/(f(X))$	Cyclotomic ring
R_q	$= \mathbb{Z}_q[X]/(f(X))$	Ring
k		The height of random matrices $\mathbf{A}$
ℓ		The width of random matrices $\mathbf{A}$
γ		Parameter defining the tail-bound of Lemma 4.2
B	$= \gamma\sigma\sqrt{N(\ell+k)}$	The maximum L^2-norm of signature share $\mathbf{z}_j \in R^{\ell+k}$ for $j \in [n]$
B_n	$= \sqrt{n}B$	The maximum L^2-norm of combined signature $\mathbf{z} \in R^{\ell+k}$
κ		The maximum L^1-norm of challenge vector c
C	$= \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\}$	Challenge space where $\|C\| = \binom{N}{\kappa}2^\kappa$
S_η	$= \{x \in R : \|x\|_\infty \leq \eta\}$	Set of small secrets
T	$= \kappa\eta\sqrt{N(\ell+k)}$	Chosen such that Lemma 4.4 holds
α		Parameter defining σ and M
σ	$= s/\sqrt{2\pi} = \alpha T$	Standard deviation of the Gaussian distribution
t	$= \omega(\sqrt{\log(mN)}) \wedge t = o(\log(mN))$	Parameter defining M such that Lemma 4.3 holds
M	$= e^{t/\alpha+1/(2\alpha^2)}$	The expected number of restarts until a single party can proceed
M_n	$= M^n$	The expected number of restarts until all n parties proceed simultaneously
cpp		Parameters for commitment scheme honestly generated with CSetup
l_1, l_2, l_4		Output bit lengths of random oracles $\mathsf{H}_1, \mathsf{H}_2$ and H_4

Table 4.2: Parameters for our distributed signature protocols.

reason, the exchange of $\mathbf{t}_j$'s is also carried with the random oracle. This way, we can prevent the adversary from choosing some malicious public key share depending on the honest party's share (the so-called *rogue key attack* [MOR01]). Furthermore, the party's index j is concatenated with the values to be hashed for the sake of "domain separation" [BDG20]. This way, we prevent rushing adversaries from simply sending back the hash coming from the honest party and claiming that they know the preimage after seeing the honest party's opening.

Signature generation. The first crucial step of $\mathsf{DS}_2.\mathsf{Sign}_n$ in Fig. 4.7 is commitment key generation at **Inputs** 3; in fact, if instead some fixed key ck was used for all signing attempts, one could come up with a sub-exponential attack that outputs a valid forgery with respect to the joint public key $\mathbf{t}$. In [DOTT20] we sketch a variant of the concurrent attack due to Drijvers et al. [DEF+19]. The original attack was against two-round Schnorr multi-signatures including BCJ scheme [BCJ08], but due to the very similar structure of FSwA-based lattice signatures an attack would become feasible against a fixed-key variant of DS_2. This motivates us to derive a message-dependent commitment key, following the mBCJ scheme of Drijvers et al.

Then the signing protocol starts by exchanging the first "commit" messages of Σ-protocol, from which all parties derive a single joint challenge $c \in C$ via a random oracle. As we discussed earlier no participants are allowed to reveal $\mathbf{w}_j$ until the rejection sampling phase, and instead they send its commitment com_j, which is to be opened only if the signature share $\mathbf{z}_j$ passes the rejection sampling. Finally, the com_j's and r_j's are added together in a meaningful way, thanks to the homomorphic property of commitment scheme.

Verification and correctness. Thanks to linearity of the underlying scheme and homomorphism of the commitment, the verifier only needs to validate the sum of signature shares, commitments and randomness. Here the Euclidean-norm bound B_n is set according to Lemma 4.1; if all parties honestly follow the protocol then the sum of n Gaussian shares

Protocol 1: $\mathsf{DS}_2.\mathsf{Gen}_n(\mathsf{pp})$

The protocol is parameterized by public parameters described in Table 4.2 and relies on the random oracles $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2}$.

Matrix Generation

1. Sample a random matrix share $\mathbf{A}_n \xleftarrow{\$} R_q^{k\times\ell}$ and generate a random oracle commitment $g_n \leftarrow \mathsf{H}_1(\mathbf{A}_n, n)$. Send out g_n.
2. Upon receiving g_j for all $j \in [n-1]$ send out $\mathbf{A}_n$.
3. Upon receiving $\mathbf{A}_j$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_1(\mathbf{A}_j, j) \neq g_j$ for some j then send out `abort`.
   b. Otherwise set public random matrix $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$, where $\mathbf{A} := \sum_{j\in[n]} \mathbf{A}_j$.

Key Pair Generation

1. Sample a secret key share $\mathbf{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$ and compute a public key share $\mathbf{t}_n := \bar{\mathbf{A}}\mathbf{s}_n$, respectively, and generate a random oracle commitment $g'_n \leftarrow \mathsf{H}_2(\mathbf{t}_n, n)$. Send out g'_n.
2. Upon receiving g'_j for all $j \in [n-1]$ send out $\mathbf{t}_n$.
3. Upon receiving $\mathbf{t}_j$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_2(\mathbf{t}_j, j) \neq g'_j$ for some j then send out `abort`.
   b. Otherwise set a combined public key $\mathbf{t} := \sum_{j\in[n]} \mathbf{t}_j$

If the protocol does not abort, P_n obtains $(\mathsf{sk}_n, \mathsf{pk}) = (\mathbf{s}_n, (\mathbf{A}, \mathbf{t}))$ as local output.

Algorithm $\mathsf{DS}_2.\mathsf{Ver}((\mathsf{com}, \mathbf{z}, r), \mu, \mathsf{pk})$

Upon receiving a message μ, signature $(\mathsf{com}, \mathbf{z}, r)$, and combined public key $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$, generate a commitment key $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$, derive a challenge $c \leftarrow \mathsf{H}_0(\mathsf{com}, \mu, \mathsf{pk})$ and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then accept if $\|\mathbf{z}\|_2 \leq B_n$ and $\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, \mathbf{w}) = 1$.

Figure 4.6: Distributed n-out-of-n signature scheme.

is only $\sqrt{n}$ times larger (while if we employed the plain Dilithium as a base scheme then the bound would grow almost linearly). Hence together with the tail-bound of Lemma 4.2 it is indeed sufficient to set $B_n = \sqrt{n}B$ for the correctness to hold with overwhelming probability. To guarantee perfect correctness, the bound check can be also carried out during the signing protocol so that it simply restarts when the generated signature is too large (which of course only happens with negligible probability and shouldn't matter in practice).

4.3.2 Asymptotic efficiency analysis

Number of aborts and signature size. As indicated in Table 4.2 the probability that all participants simultaneously proceed is $1/M_n = 1/M^n$, where $1/M$ is the probability that each party asks to proceed. To make M_n reasonably small, say $M_n = 3$, we should set $\alpha \geq 11n$ [DLL+18], leading to $\sigma \geq 11nT$. This already increases the bound B of

Protocol 2: $\mathsf{DS}_2.\mathsf{Sign}_n(\mathsf{sid}, \mathsf{sk}_n, \mathsf{pk}, \mu)$

The protocol is parameterized by public parameters described in Table 4.2 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{\mathsf{ck}}$. The protocol assumes that $\mathsf{DS}_2.\mathsf{Gen}_n(\mathsf{pp})$ has been previously invoked. If a party halts with `abort` at any point, then all $\mathsf{Sign}_n(\mathsf{sid}, \mathsf{sk}_n, \mathsf{pk}, \mu)$ executions are aborted.

Inputs

1. P_n receives a unique session ID sid, $\mathsf{sk}_n = \mathbf{s}_n$, $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and message $\mu \in \{0,1\}^*$ as input.
2. P_n verifies that sid has not been used before (if it has been, the protocol is not executed).
3. P_n locally computes a per-message commitment key $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$.

Signature Generation P_n works as follows:

1. Compute the first message as follows.
   a. Sample $\mathbf{y}_n \xleftarrow{\$} D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   b. Compute $\mathsf{com}_n \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathbf{w}_n; r_n)$ with $r_n \xleftarrow{\$} D(S_r)$.
   c. Send out com_n.
2. Upon receiving com_j for all $j \in [n-1]$ compute the signature share as follows.
   a. Set $\mathsf{com} := \sum_{j\in[n]} \mathsf{com}_j$.
   b. Derive a challenge $c \leftarrow \mathsf{H}_0(\mathsf{com}, \mu, \mathsf{pk})$.
   c. Computes a signature share $\mathbf{z}_n := c\mathbf{s}_n + \mathbf{y}_n$.
   d. Run the rejection sampling on input $(c\mathbf{s}_n, \mathbf{z}_n)$, i.e., with probability
   $$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$
   send out $(\mathbf{z}_n, r_n)$; otherwise send out `restart` and go to 1.
3. Upon receiving `restart` from some party go to 1. Otherwise upon receiving $(\mathbf{z}_j, r_j)$ for all $j \in [n-1]$ compute the combined signature as follows
   a. For each $j \in [n-1]$ reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ and validate the signature share:
   $$\|\mathbf{z}_j\|_2 \le B \text{ and } \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out `abort`.
   b. Compute $\mathbf{z} := \sum_{j\in[n]} \mathbf{z}_j$ and $r := \sum_{j\in[n]} r_j$.

If the protocol does not abort, P_n obtains a signature $(\mathsf{com}, \mathbf{z}, r)$ as local output.

Figure 4.7: Distributed n-out-of-n signature scheme.

each signature share linearly compared to a non-distributed signature like Dilithium-G. In addition, we should set the bound B_n for combined signature to $\sqrt{n}B$ for the correctness to hold, and thus the SIS solution that we find in the security reduction grows by a factor of $n^{3/2}$.

This translates to a signature size increase of a factor of roughly[4] $O(\log n)$, so the scaling in terms of the number of parties is reasonable. In addition, when using the trapdoor commitment scheme of Section 4.5, one can substantially reduce the signature size by using the common Fiat–Shamir trick of expressing the signature as $(c, \mathbf{z}, r)$ instead of $(\mathsf{com}, \mathbf{z}, r)$, and carrying out the verification by first recomputing the commitment using the randomness r, and then checking the consistency of the challenge: $c \stackrel{?}{=} \mathsf{H}_0(\mathsf{com}, \mu, \mathsf{pk})$. This makes signature size somewhat closer to the original Dilithium-G, although commitment randomness r may be still relatively large compared to $\mathbf{z}$ depending on the instantiation of TCOM (see Section 4.5.2).

We expect that a number of further optimizations are possible to improve the efficiency of this protocol in both asymptotic and concrete terms (e.g., by relying on stronger assumptions like (Mod-)NTRU), although this is left for further work. Accordingly, we also leave for further work the question of providing concrete parameters for the protocol, since the methodology for setting parameters is currently a moving target (e.g., the original parameters for Dilithium-G are not considered up-to-date), there is arguably no good point of comparison in the literature (in particular, no previous lattice-based two-round protocol), and again, a concrete instantiation would likely rely on stronger assumptions to achieve better efficiency anyway.

Round complexity. If this protocol is used as it is, it only outputs a signature after the three rounds with probability $1/M_n$ (which is $1/3$ with the parameters above). As a result, to effectively compute a signature, it has to be repeated M_n times on average, and so the expected number of rounds is in fact larger than 2 ($2M_n = 6$ in this case). One can of course adjust the parameters to reduce M_n to any constant greater than 1, or even to $1 + o(1)$ by picking e.g. $\alpha = \Theta(n^{1+\epsilon})$; this results in an expected number of rounds arbitrarily close to 2. Alternatively, one can keep a 2-round protocol while ensuring that the parties output a signature with overwhelming probability, simply by running sufficiently many parallel executions of the protocol at once: since the probability that all τ parallel executions simultaneously restart is given by $2^{-\lambda} = (1 - 1/M_n)^\tau$, we find that $\tau = \lambda / \left(\log \frac{M_n}{M_n - 1}\right)$ parallel executions suffice if λ is the security parameter.

4.3.3 Security

The formal security claim for our DS_2 protocol is given below.

Theorem 4.1. *Suppose the trapdoor commitment scheme* TCOM *is secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that initiates a single key generation protocol by querying* $\mathcal{O}_n^{\mathsf{DS}_2}$ *with* $\mathsf{sid} = 0$*, initiates* Q_s

[4] To be more precise, since the verification bound scales as $n^{3/2}$, one should also increase q by the same bound to avoid arithmetic overflow. This makes the MSIS problem harder, but the MLWE easier if the dimension is kept unchanged. To keep the same security level, one should therefore also increase N by a factor of $1 + O(\frac{\log n}{\log q_0})$ where q_0 is the value of q in the single-user setting. Therefore, one could in principle argue that signature size actually scales as $O(\log^2 n)$. However, one typically chooses $q_0 > 2^{20}$, and therefore even in settings with billions of parties, $\frac{\log n}{\log q_0} < 2$. Thus, one can effectively regard N as independent of n.

signature generation protocols by querying $\mathcal{O}_n^{\mathsf{DS}_2}$ *with* $\mathsf{sid} \neq 0$*, and makes* Q_h *queries to the random oracle* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$*, the protocol* DS_2 *of Fig. 4.7 is* DS-UF-CMA *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions, where* $\beta = 2\sqrt{B_n^2 + \kappa}$*. Concretely, using other parameters specified in Table 4.2, the advantage of* $\mathcal{A}$ *is bounded as follows.*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}_2}(\mathcal{A}) \leq e \cdot (Q_h + Q_s + 1) \cdot \Bigg(& (Q_h + Q_s)\epsilon_{td} + Q_s \cdot \frac{2e^{-t^2/2}}{M} + \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}} \\ & + \frac{(Q_h+1)Q_h}{2^{l_1+1}} + \frac{Q_h}{q^{k\ell N}} + \frac{n}{2^{l_1}} + \frac{(Q_h+1)Q_h}{2^{l_2+1}} + \frac{Q_h}{q^{kN}} + \frac{n}{2^{l_2}} \\ & + \frac{Q_h + Q_s + 1}{|C|} + \sqrt{(Q_h + Q_s + 1) \cdot \left(\epsilon_{\mathsf{bind}} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}\right)} \Bigg)\end{aligned}$$

We first give a sketch of the security proof. We remark that its multi-signature variant MS_2 (Section 4.4) can be proven secure relying on essentially the same idea. We show that given any efficient adversary $\mathcal{A}$ that creates a valid forgery with non-negligible probability, one can break either $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ assumption or computational binding of TCOM.

Key generation simulation. For the key generation phase, since the public key share of the honest signer $\mathbf{t}_n$ is indistinguishable from the vector sampled from R_q^k uniformly at random due to $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption, the honest party oracle simulator can replace $\mathbf{t}_n$ with such a vector. Therefore, the distribution of combined public key $\mathbf{t} = \sum_{j\in[n]} \mathbf{t}_j$ is also indistinguishable from the uniform distribution. Thanks to the random oracle commitment, after the adversary has submitted g'_j for each $j \in [n-1]$ one can extract the adversary's public key share $\mathbf{t}_j$, with which the simulator sets its share a posteriori $\mathbf{t}_n := \mathbf{t} - \sum_{j\in[n-1]} \mathbf{t}_j$ and programs the random oracle accordingly $\mathsf{H}_2(\mathbf{t}_n, n) := g'_n$. Using the same argument, one can set a random matrix share $\mathbf{A}_n := \mathbf{A} - \sum_{j\in[n-1]} \mathbf{A}_j$ given a resulting random matrix $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$. Now we can embed an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \xleftarrow{\$} R_q^{k\times(\ell+1)}$. Due to the way we simulated the joint public key $(\mathbf{A}, \mathbf{t})$ is uniformly distributed in $R_q^{k\times\ell} \times R_q^k$, so replacing it with a $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ instance doesn't change the view of adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$.

Signature generation simulation. The oracle simulation closely follows the one for mBCJ [DEF+19]. Concretely, the oracle simulator programs H_3 so that for each signing query it returns tck generated via $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{TCGen}(\mathsf{cpp})$, and for the specific crucial query that is used to create a forgery it returns an actual commitment key $\mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp})$, which has been received by the reduction algorithm as a problem instance of the binding game. This way, upon receiving signing queries the oracle simulator can send out a "fake" commitment $\mathsf{com}_n \leftarrow \mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$ at the first round, and then the known trapdoor td allows to later equivocate to a simulated first message of the Σ-protocol *after* the joint random challenge $c \in C$ has been derived; formally, it samples a simulated signature share $\mathbf{z}_n \xleftarrow{\$} D_s^{\ell+k}$ and then derives randomness as $r_n \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_n, \mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$. On the other hand, when the reduction obtains two openings after applying the forking lemma it can indeed break the binding property with respect to a real commitment key ck.

Forking lemma. Our proof is relying on the forking lemma [PS00a, BN06]. This is mainly because we instantiated the protocol with a trapdoor commitment, which inevitably implies that the binding is only computational. Hence to construct a reduction that breaks binding, we do have to make the adversary submit two valid openings for a single commitment

under the same key, which seems to require some kind of rewinding technique. After applying the forking lemma, the adversary submits two forgeries with distinct challenges $c^* \neq \hat{c}^*$, with which we can indeed find a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, or otherwise break computational binding with respect to ck. Concretely, after invoking the forking lemma, we obtain two forgeries $(\mathsf{com}^*, \mathbf{z}^*, r^*, \mu^*)$ and $(\hat{\mathsf{com}}^*, \hat{\mathbf{z}}^*, \hat{r}^*, \hat{\mu}^*)$ such that $c^* = \mathsf{H}(\mathsf{com}^*, \mu, \mathsf{pk}) \neq \mathsf{H}(\hat{\mathsf{com}}^*, \hat{\mu}^*, \mathsf{pk}) = \hat{c}^*$, $\mathsf{com}^* = \hat{\mathsf{com}}^*$, $\mu^* = \hat{\mu}^*$, and $\mathsf{H}(\mu^*, \mathsf{pk}) = \mathsf{H}(\hat{\mu}^*, \mathsf{pk}) = \mathsf{ck}$. Since both forgeries are verified, we have $\|\mathbf{z}^*\|_2 \leq B_n \wedge \|\hat{\mathbf{z}}^*\|_2 \leq B_n$, and

$$\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}^*, r^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}) = \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}^*, \hat{r}^*, \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}) = 1.$$

If $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} \neq \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$ then it means that computational binding is broken with respect to a commitment key ck. Suppose $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} = \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$. Rearranging it leads to

$$[\mathbf{A}|\mathbf{I}|\mathbf{t}] \begin{bmatrix} \mathbf{z}^* - \hat{\mathbf{z}}^* \\ \hat{c}^* - c^* \end{bmatrix} = \mathbf{0}.$$

Recalling that $[\mathbf{A}'|\mathbf{I}] = [\mathbf{A}|\mathbf{t}|\mathbf{I}]$ is an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem, we have found a valid solution if $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $\|\mathbf{z}^* - \hat{\mathbf{z}}^*\|_2 \leq 2B_n$ and $0 < \|\hat{c}^* - c^*\|_2 \leq \sqrt{4\kappa}$.

Full security proof.

Proof. Given $\mathcal{A}$ against DS_2 we show that its advantage $\mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}_2}(\mathcal{A})$ is negligible by constructing a reduction. Without loss of generality we assume that P_n is an honest party. Our first goal is to construct an algorithm $\mathcal{B}$ around $\mathcal{A}$ that simulates the behaviors of P_n without using honestly generated key pairs. Then we invoke the forking algorithm $\mathcal{F}_{\mathcal{B}}$ from Lemma 4.5 to obtain two forgeries with distinct challenges, which allow to construct a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ or to break computational binding of commitment scheme TCOM. We present the resulting $\mathcal{B}$ in Fig. 4.8, together with its subroutines Figs. 4.9 to 4.13. Below we discuss how to realize this via several intermediate hybrids.

G_0 **Random oracle simulation.** We assume that $\mathcal{B}$ receives random samples $h_i \xleftarrow{\$} C$ for each $i \in [Q_h + Q_s + 1]$ as input. The random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$, $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2}$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{\mathsf{ck}}$ are simulated as follows. The table HT_i is initially empty. The $\mathcal{B}$ also maintains a counter ctr which is initially set to 0. Note that the slightly involved simulation of H_0 below will come into play when the forking lemma is applied; this way, the adversary $\mathcal{A}$'s view is indeed identical until the forking point in two executions.

$\mathsf{H}_0(x)$ 1. Parse x as $(\mathsf{com}, \mu, \mathsf{pk})$; 2. Make a query $\mathsf{H}_3(\mu, \mathsf{pk})$, so that $\mathrm{HT}_3[\mu, \mathsf{pk}]$ is immediately set; 3. If $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}] = \bot$ then increment ctr and set $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}] := h_{\mathsf{ctr}}$; 4. Return $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}]$.

$\mathsf{H}_1(x)$ If $\mathrm{HT}_1[x]$ is not set let $\mathrm{HT}_1[x] \xleftarrow{\$} \{0,1\}^{l_1}$. Return $\mathrm{HT}_1[x]$.

$\mathsf{H}_2(x)$ If $\mathrm{HT}_2[x]$ is not set let $\mathrm{HT}_2[x] \xleftarrow{\$} \{0,1\}^{l_2}$. Return $\mathrm{HT}_2[x]$.

$\mathsf{H}_3(\mu, \mathsf{pk})$ If $\mathrm{HT}_3[\mu, \mathsf{pk}]$ is not set let $\mathrm{HT}_3[\mu, \mathsf{pk}] \xleftarrow{\$} S_{\mathsf{ck}}$. Return $\mathrm{HT}_3[\mu, \mathsf{pk}]$.

Honest party oracle simulation. In this game $\mathcal{B}$ behaves exactly like a single honest party in DS_2; concretely, it simulates an oracle $\mathcal{O}_n^{\mathsf{DS}_2}$ (Fig. 4.3) which internally invokes instructions of Gen_n and Sign_n according to Fig. 4.7, respectively.

Forgery. When $\mathcal{A}$ outputs a forgery $(\mathsf{com}^*, \mathbf{z}^*, r^*, \mu^*)$ at the end $\mathcal{B}$ proceeds as follows.

1. If $\mu^* \in \mathit{Mset}$ then $\mathcal{B}$ halts with output $(0, \perp)$
2. Make queries $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathsf{pk})$ and $c^* \leftarrow \mathsf{H}_0(\mathsf{com}^*, \mu^*, \mathsf{pk})$
3. If $\mathsf{Open}_{\mathsf{ck}^*}(\mathsf{com}^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}, r^*) \neq 1$ or $\|\mathbf{z}^*\|_2 > B_n$ then $\mathcal{B}$ halts with output $(0, \perp)$.
4. Find $i_{\mathrm{f}} \in [Q_h + Q_s + 1]$ such that $c^* = h_{i_{\mathrm{f}}}$ and $\mathcal{B}$ halts with output $(i_{\mathrm{f}}, \mathit{out} = (\mathsf{com}^*, c^*, \mathbf{z}^*, r^*, \mu^*, \mathsf{ck}^*))$

Let $\Pr[\mathrm{G}_i]$ denote a probability that $\mathcal{B}$ doesn't output $(0, \perp)$ at the game G_i. Then we have

$$\Pr[\mathrm{G}_0] = \mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}_2}(\mathcal{A}).$$

G_1 This game is identical to G_0 except at the following points.

Random oracle simulation. Simulation of the random oracle H_3 is analogous to Drijvers et al. [DEF+19] The core idea is to make sure that all sign queries can be responded with trapdoor commitments, which can be equivocated to an arbitrary plaintext later, and that the forgery submitted by $\mathcal{A}$ involves the actual commitment key. In this game $\mathcal{B}$ initially generates a single commitment key $\mathsf{ck} \xleftarrow{\$} S_{\mathsf{ck}}$. Then upon receiving a query (μ, pk) to H_3, $\mathcal{B}$ tosses a biased coin that comes out heads with probability ϖ and tails with $1 - \varpi$. If the coin comes out heads, then $\mathcal{B}$ invokes TCGen to generate a commitment key–trapdoor pair $(\mathsf{tck}, \mathsf{td})$, stores tck and td in tables $\mathrm{HT}_3[\mu, \mathsf{pk}]$ and $\mathrm{TDT}[\mu, \mathsf{pk}]$, respectively, and returns tck; if it comes out tails, then $\mathcal{B}$ stores a predefined ck in $\mathrm{HT}_3[\mu, \mathsf{pk}]$ and returns ck. The complete description of random oracle simulation is presented in Fig. 4.10.

Honest party oracle simulation. The $\mathcal{B}$ differs from the prior one at the following steps in $\mathsf{DS}_2.\mathsf{Sign}_n$.

> **Inputs** 3 Call $\mathsf{H}_3(\mu, \mathsf{pk})$ to obtain tck. If $\mathrm{TDT}[\mu, \mathsf{pk}] = \perp$ (i.e., TCGen was not called) then set a flag bad_4 and halt with output $(0, \perp)$. Otherwise obtain the trapdoor $\mathsf{td} \leftarrow \mathrm{TDT}[\mu, \mathsf{pk}]$
>
> **Signature Generation** 1.b. Call $\mathsf{com}_n \leftarrow \mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$ instead of committing to $\mathbf{w}_n$.
>
> **Signature Generation** 2.c. After computing $\mathbf{z}_n := c\mathbf{s}_n + \mathbf{y}_n$ derive randomness $r_n \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_n, \mathbf{w}_n)$.

Forgery. When $\mathcal{A}$ outputs a successful forgery $(\mathsf{com}^*, \mathbf{z}^*, r^*, \mu^*)$ at the end, we modify the step 3 of G_0 as follows.

> **Forgery** 3 If $\mathsf{Open}_{\mathsf{ck}^*}(\mathsf{com}^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}, r^*) \neq 1$ or $\|\mathbf{z}^*\|_2 > B_n$ then $\mathcal{B}$ halts with output $(0, \perp)$. *If* $\mathrm{TDT}[\mu^*, \mathsf{pk}] \neq \perp$ *(i.e.,* TCGen *was called for a query* $\mathsf{H}_3(\mu^*, \mathsf{pk})$*)* then set bad_5 and $\mathcal{B}$ halts with output $(0, \perp)$.

Note that due to the way H_3 is simulated, if $\mathcal{B}$ does not output $(0, \perp)$ it is now guaranteed that $\mathsf{ck}^* = \mathsf{ck} = \mathsf{H}_3(\mu^*, \mathsf{pk})$. Recalling that TCOM is secure (Definition 4.4) we have

$$\Pr[\mathrm{G}_1] \geq \varpi^{Q_h + Q_s} \cdot (1 - \varpi) \cdot \Pr[\mathrm{G}_0] - (Q_h + Q_s) \cdot \epsilon_{\mathrm{td}}$$

because the simulation is only successful if the random oracle H_3 internally uses TCGen for all but one queries to H_3 (both directly and indirectly via H_0 and Sign_n) *and* if H_3 uses a predefined ck for a single crucial query (μ^*, pk) associated with

forgery; in other words, it is only successful if neither bad_4 nor bad_5 is set above. Note that by setting $\varpi = (Q_h+Q_s)/(Q_h+Q_s+1)$ since $(1/(1+1/(Q_h+Q_s)))^{Q_h+Q_s} \geq 1/e$ for $Q_h + Q_s \geq 0$ we obtain

$$\Pr[\mathrm{G}_1] \geq \frac{\Pr[\mathrm{G}_0]}{e(Q_h+Q_s+1)} - (Q_h+Q_s)\cdot\epsilon_{\mathrm{td}}.$$

G_2 This game is identical to G_1 except at the following points.

Honest party oracle simulation. The $\mathcal{B}$ doesn't honestly generate $\mathbf{z}_n$ anymore and instead simulates the rejection sampling as follows.

Signature Generation 1.a. $\mathcal{B}$ does nothing here.

Signature Generation 2.c. Sample $\mathbf{z}_n \xleftarrow{\$} D_s^{\ell+k}$ and derive randomness $r_n \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_n, \mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$.

Signature Generation 2.d. With probability $1/M$ send out $(\mathbf{z}_n, r_n)$. Otherwise send out `restart`.

The signature share $\mathbf{z}_n$ simulated this way is statistically indistinguishable from the real one because of special HVZK property of the underlying identification scheme. In other words, we can directly apply the result of Lemmas 4.3 and 4.4. Hence we have

$$|\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_1]| \leq Q_s \cdot \frac{2e^{-t^2/2}}{M}.$$

G_3 At this stage, notice that the signing queries are simulated according to $\mathsf{SimSign}_n$ in Fig. 4.13, and it doesn't rely on the actual secret key share $\mathbf{s}_n$ anymore. So our next goal is to simulate the generation of $\mathbf{t}_n$ without using $\mathbf{s}_n$. In this game $\mathcal{B}$ first picks the resulting random matrix $\mathbf{A} \in R_q^{k\times\ell}$ and defines its own share $\mathbf{A}_n$ a posteriori, after extracting adversary's committed shares $\mathbf{A}_1, \ldots, \mathbf{A}_{n-1}$. This can be done by searching the recorded random oracle queries in HT_1. Note that the distributions of $\mathbf{A}$ and $\mathbf{A}_n$ haven't changed from the previous game. The formal simulation strategy is described in **Matrix Generation** section of Fig. 4.12. Since G_3 is identical to G_2 from adversary $\mathcal{A}$'s point of view except at the bad events marked there, we have

$$|\Pr[\mathrm{G}_3] - \Pr[\mathrm{G}_2]| \leq \Pr[\mathsf{bad}_1] + \Pr[\mathsf{bad}_2] + \Pr[\mathsf{bad}_3] \leq \frac{(Q_h+1)Q_h}{2^{l_1+1}} + \frac{Q_h}{q^{k\ell N}} + \frac{n}{2^{l_1}}$$

where $\Pr[\mathsf{bad}_1]$ corresponds to the probability that at least one collision occurs during at most Q_h queries to H_1 made by $\mathcal{A}$ or $\mathcal{B}$, which is at most $((Q_h+1)Q_h/2)/2^{l_1}$; $\Pr[\mathsf{bad}_2]$ is the probability that programming the random oracle H_1 fails, which happens only if $\mathsf{H}_1(\mathbf{A}_n, n)$ has been previously asked by $\mathcal{A}$ during at most Q_h queries to H_1, and the probability that guessing a uniformly random $\mathbf{A}_n$ by chance is at most $1/q^{k\ell N}$ for each query; $\Pr[\mathsf{bad}_3]$ is the probability that $\mathcal{A}$ has predicted one of the $n-1$ outputs of random oracle H_1 without making a query to it, which could only happen with probability at most $n/2^{l_1}$.

G_4 This game is identical to G_3 except that $\mathcal{B}$ simply picks the random public key share $\mathbf{t}_n \xleftarrow{\$} R_q^k$ during the key generation phase, instead of computing $\mathbf{t}_n = \bar{\mathbf{A}}\mathbf{s}_n$ where $\mathbf{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$. As $\mathbf{A}$ follows the uniform distribution over $R_q^{k\times\ell}$, if the adversary $\mathcal{A}$ can distinguish G_3 and G_4 then we can use $\mathcal{A}$ as a distinguisher that breaks $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption; hence we have

$$|\Pr[\mathrm{G}_4] - \Pr[\mathrm{G}_3]| \leq \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}}.$$

G_5 In this game $\mathcal{B}$ first picks the resulting public key $\mathbf{t}$ randomly from R_q^k and defines its own share $\mathbf{t}_n$ a posteriori, after extracting adversary's committed shares $\mathbf{t}_1, \ldots, \mathbf{t}_{n-1}$. This can be done by searching the recorded random oracle queries in HT_2. Note that the distributions of $\mathbf{t}$ and $\mathbf{t}_n$ haven't changed from the previous game. The formal simulation strategy is described in **Key Pair Generation** section of Fig. 4.12. Since G_5 is identical to G_4 from adversary $\mathcal{A}$'s point of view except at the bad' events marked there, we have

$$|\Pr[\mathrm{G}_5] - \Pr[\mathrm{G}_4]| \leq \Pr[\mathsf{bad}'_1] + \Pr[\mathsf{bad}'_2] + \Pr[\mathsf{bad}'_3] \leq \frac{(Q_h+1)Q_h}{2^{l_2+1}} + \frac{Q_h}{q^{kN}} + \frac{n}{2^{l_2}}$$

where the bounds are calculated just as in G_3.

Forking lemma. At this stage, notice that the key generation query is simulated according to SimGen_n in Fig. 4.12. Our goal is to embed a challenge commitment key $\mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp})$ and an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \xleftarrow{\$} R_q^{k\times(\ell+1)}$. As in G_5 the combined public key $(\mathbf{A}, \mathbf{t})$ is uniformly distributed in $R_q^{k\times\ell} \times R_q^k$, replacing it with $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ instance doesn't change the view of adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$. Moreover, thanks to the simulation of H_3 it is guaranteed that ck follows the uniform distribution over S_{ck} which is perfectly indistinguishable from honestly generated $\mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp})$ (since the keys are uniform). Hence we define the input generator IGen of forking lemma such that it outputs the instance $(\mathsf{ck}, \mathbf{A}, \mathbf{t})$.

Now we prove the theorem by constructing $\mathcal{B}'$ around $\mathcal{B}$ in Fig. 4.8 that either (1) breaks binding of commitment with respect to ck, or (2) finds a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ on input $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$. The $\mathcal{B}'$ invokes the forking algorithm $\mathcal{F}_{\mathcal{B}}$ on input $(\mathsf{ck}, \mathbf{A}, \mathbf{t})$ from Lemma 4.5. Then with probability frk we immediately get two forgeries $out = (\mathsf{com}^*, c^*, \mathbf{z}^*, r^*, \mu^*, \mathsf{ck}^*)$ and $\hat{out} = (\hat{\mathsf{com}}^*, \hat{c}^*, \hat{\mathbf{z}}^*, \hat{r}^*, \hat{\mu}^*, \hat{\mathsf{ck}}^*)$, where frk satisfies

$$\Pr[\mathrm{G}_5] = \mathrm{acc} \leq \frac{Q_h + Q_s + 1}{|C|} + \sqrt{(Q_h + Q_s + 1) \cdot \mathrm{frk}}.$$

By construction of $\mathcal{B}$ and $\mathcal{F}_{\mathcal{B}}$ we have $\mathsf{com}^* = \hat{\mathsf{com}}^*$, $\mu^* = \hat{\mu}^*$ and $c^* \neq \hat{c}^*$; until the forking point $\mathsf{ctr} = i_{\mathrm{f}}$ the adversary $\mathcal{A}$'s view is identical in two executions. Moreover, due to the simulation of H_0 we also guarantee that $\mathsf{ck}^* = \hat{\mathsf{ck}}^* = \mathsf{ck}$ since $\mathsf{H}_3(\mu^*, \mathsf{pk})$ and $\mathsf{H}_3(\hat{\mu}^*, \mathsf{pk})$ should have been invoked right before the fork. Since both forgeries are verified under the same commitment key ck, we have $\|\mathbf{z}^*\|_2 \leq B_n \wedge \|\hat{\mathbf{z}}^*\|_2 \leq B_n$ and

$$\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}^*, r^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}) = \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}^*, \hat{r}^*, \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}) = 1.$$

If $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} \neq \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$ then $\mathcal{B}'$ can break computational binding with respect to ck, which succeeds with probability at most ϵ_{bind}. If $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} = \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$, rearranging it leads to

$$[\mathbf{A}|\mathbf{I}|\mathbf{t}] \begin{bmatrix} \mathbf{z}^* - \hat{\mathbf{z}}^* \\ \hat{c}^* - c^* \end{bmatrix} = \mathbf{0}.$$

Recalling that $[\mathbf{A}'|\mathbf{I}] = [\mathbf{A}|\mathbf{t}|\mathbf{I}]$ is an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem, we have found a valid solution if $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $\|\mathbf{z}^* - \hat{\mathbf{z}}^*\|_2 \leq 2B_n$ and $0 < \|\hat{c}^* - c^*\|_2 \leq \sqrt{4\kappa}$. Putting two cases together, we get

$$\mathrm{frk} \leq \epsilon_{\mathsf{bind}} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}.$$

□

Algorithm $\mathcal{B}((\mathsf{ck}, \mathbf{A}, \mathbf{t}), h_1, \ldots, h_{Q_h+Q_s+1})$

The algorithm is initialized with empty hash tables HT_i for $i = 0, \ldots, 3$, trapdoor table TDT, a set of queried messages $Mset = \varnothing$, and a counter $\mathsf{ctr} = 0$.

Honest party oracle simulation Upon receiving a query of the form (sid, m) from $\mathcal{A}$, reply the query as described in $\mathsf{Sim}\mathcal{O}_n^{\mathsf{DS}_2}(\mathsf{sid}, m)$ (Fig. 4.9). If $\mathsf{Sim}\mathcal{O}_n^{\mathsf{DS}_2}$ halts with output $(0, \perp)$ then $\mathcal{B}$ also halts with output $(0, \perp)$.

Random oracle simulation Upon receiving a query to the random oracles from $\mathcal{A}$, reply the query as described in Fig. 4.10.

Forgery Upon receiving a forgery $(\mathsf{com}^*, \mathbf{z}^*, r^*, \mu^*)$ from $\mathcal{A}$:

1. If $\mu^* \in Mset$ then $\mathcal{B}$ halts with output $(0, \perp)$
2. Make queries $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathsf{pk})$ and $c^* \leftarrow \mathsf{H}_0(\mathsf{com}^*, \mu^*, \mathsf{pk})$
3. If $\mathsf{Open}_{\mathsf{ck}^*}(\mathsf{com}^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}, r^*) \neq 1$ or $\|\mathbf{z}^*\|_2 > B_n$ then $\mathcal{B}$ halts with output $(0, \perp)$. If $\mathrm{TDT}[\mu^*, \mathsf{pk}] \neq \perp$ (i.e., TCGen was called for a query $\mathsf{H}_3(\mu^*, \mathsf{pk})$) then set bad_5 and $\mathcal{B}$ halts with output $(0, \perp)$.
4. Find $i_{\mathrm{f}} \in [Q_h + Q_s + 1]$ such that $c^* = h_{i_{\mathrm{f}}}$ and $\mathcal{B}$ halts with output $(i_{\mathrm{f}}, out = (\mathsf{com}^*, c^*, \mathbf{z}^*, r^*, \mu^*, \mathsf{ck}^*))$

Figure 4.8: The algorithm simulating the view of $\mathcal{A}$ in $\mathsf{Exp}_{\mathsf{DS}_2}^{\mathsf{DS\text{-}UF\text{-}CMA}}(\mathcal{A})$ experiment

Oracle $\mathsf{Sim}\mathcal{O}_n^{\mathsf{DS}_2}(\mathsf{sid}, m)$

The simulator is initialized with public parameters pp generated by Setup algorithm. The variable flag is initially set to false.

Key Generation Upon receiving $(0, m)$, if $\mathsf{flag} = \mathsf{true}$ then return $\perp$. Otherwise do the following:

- If the oracle is queried with $\mathsf{sid} = 0$ for the first time then it initializes a machine $\mathcal{M}_0$ running the instructions $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \mathbf{t})$ (Fig. 4.12). If P_n sends the first message in the key generation protocol, then this message is the oracle reply.
- If $\mathcal{M}_0$ has been already initialized then the oracle hands the machine $\mathcal{M}_0$ the next incoming message m and returns $\mathcal{M}_0$'s reply. If $\mathcal{M}_0$ fails with output $(0, \perp)$ at any point then the oracle stops the simulation with output $(0, \perp)$. If $\mathcal{M}_0$ concludes $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \mathbf{t})$ with local output $(\mathbf{t}_n, \mathsf{pk})$, then set $\mathsf{flag} = \mathsf{true}$.

Signature Generation Upon receiving (sid, m) with $\mathsf{sid} \neq 0$, if $\mathsf{flag} = \mathsf{false}$ then return $\perp$. Otherwise do the following:

- If the oracle is queried with sid for the first time then parse the incoming message m as μ. It initializes a machine $\mathcal{M}_{\mathsf{sid}}$ running the instructions of $\mathsf{SimSign}_n(\mathsf{sid}, \mathbf{t}_n, \mathsf{pk}, \mu)$ (Fig. 4.13). The machine $\mathcal{M}_{\mathsf{sid}}$ is initialized with the key share and any state information stored by $\mathcal{M}_0$. The message μ to be signed is included in $Mset$. If P_n sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{\mathsf{sid}}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{\mathsf{sid}}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{\mathsf{sid}}$. If $\mathcal{M}_{\mathsf{sid}}$ fails with output $(0, \perp)$ at any point then the oracle stops the simulation with output $(0, \perp)$. If $\mathcal{M}_{\mathsf{sid}}$ concludes with local output σ, then the output obtained by $\mathcal{M}_{\mathsf{sid}}$ is returned.

Figure 4.9: Honest party oracle simulator for DS_2.

Algorithm Random Oracle Simulation

$\underline{\mathsf{H}_0(x)}$

1. Parse x as $(\mathsf{com}, \mu, \mathsf{pk})$
2. Make a query $\mathsf{H}_3(\mu, \mathsf{pk})$
3. If $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}] = \perp$ then increment ctr and set $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}] \coloneqq h_{\mathsf{ctr}}$
4. Return $\mathrm{HT}_0[\mathsf{com}, \mu, \mathsf{pk}]$

$\underline{\mathsf{H}_1(x)}$

1. If $\mathrm{HT}_1[x] = \perp$ then set $\mathrm{HT}_1[x] \xleftarrow{\$} \{0,1\}^{l_1}$
2. Return $\mathrm{HT}_1[x]$

$\underline{\mathsf{H}_2(x)}$

1. If $\mathrm{HT}_2[x] = \perp$ then set $\mathrm{HT}_2[x] \xleftarrow{\$} \{0,1\}^{l_2}$
2. Return $\mathrm{HT}_2[x]$

$\underline{\mathsf{H}_3(x)}$

1. Parse x as (μ, pk)
2. If $\mathrm{HT}_3[\mu, \mathsf{pk}] = \perp$:
   - With probability ϖ, compute $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{TCGen}(\mathsf{cpp})$, store the trapdoor in $\mathrm{TDT}[\mu, \mathsf{pk}] \coloneqq \mathsf{td}$ and set $\mathrm{HT}_3[\mu, \mathsf{pk}] \coloneqq \mathsf{tck}$.
   - Otherwise, set $\mathrm{HT}_3[\mu, \mathsf{pk}] \coloneqq \mathsf{ck}$.
3. Return $\mathrm{HT}_3[\mu, \mathsf{pk}]$

Figure 4.10: Random oracle simulator for DS_2.

Algorithm SearchHashTable

Upon receiving the hash table HT together with hash values $(h_1, \ldots, h_{n-1})$:

1. If for some $j \in [n-1]$ the preimage of h_j doesn't exist in HT then set the flag alert.
2. If for some $j \in [n-1]$ more than one preimages of h_j exist in HT then set the flag bad.
3. Return $(\mathsf{alert}, \mathsf{bad}, m_1, \ldots, m_{n-1})$, where for each $j \in [n-1]$ if there is no m_j such that $\mathrm{HT}[m_j] = h_j$ then $m_j = \perp$, and otherwise m_j is defined such that $\mathrm{HT}[m_j] = h_j$.

Figure 4.11: Routine for searching hash tables.

Algorithm $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \mathbf{t})$

Matrix Generation

1. Sample $g_n \xleftarrow{\$} \{0,1\}^{l_1}$ and send out g_n.
2. Upon receiving g_j for all $j \in [n-1]$ proceed as follows:
   a. Invoke $\mathsf{SearchHashTable}$ in Fig. 4.11 on input HT_1 and $(g_1, \ldots, g_{n-1})$ to obtain $(\mathsf{alert}, \mathsf{bad}_1, (\mathbf{A}_1, 1), \ldots, (\mathbf{A}_{n-1}, n-1))$.
   b. If the flag bad_1 is set then simulation fails with output $(0, \perp)$.
   c. If the flag alert is set then pick $\mathbf{A}_n \xleftarrow{\$} R_q^{k \times \ell}$. Otherwise using a predefined matrix $\mathbf{A}$ define $\mathbf{A}_n := \mathbf{A} - \sum_{j=1}^{n-1} \mathbf{A}_j$.
      - If $\mathrm{HT}_1[\mathbf{A}_n, n]$ has been already set then set bad_2 and simulation fails with output $(0, \perp)$.
      - Otherwise program the random oracle $\mathrm{HT}_1[\mathbf{A}_n, n] := g_n$ and send out $\mathbf{A}_n$.
3. Upon receiving $\mathbf{A}_j$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_1(\mathbf{A}_j, j) \neq g_j$ for some j then send out $\mathtt{abort}$.
   b. If alert is set and $\mathsf{H}_1(\mathbf{A}_j, j) = g_j$ for all j then set bad_3 and simulation fails with output $(0, \perp)$.
   c. Otherwise set a public random matrix $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$.

Key Pair Generation

1. Sample $g'_n \xleftarrow{\$} \{0,1\}^{l_2}$ and send out g'_n.
2. Upon receiving g'_j for all $j \in [n-1]$ proceed as follows:
   a. Invoke $\mathsf{SearchHashTable}$ in Fig. 4.11 on input HT_2 and $(g'_1, \ldots, g'_{n-1})$ to obtain $(\mathsf{alert}', \mathsf{bad}'_1, (\mathbf{t}_1, 1), \ldots, (\mathbf{t}_{n-1}, n-1))$.
   b. If the flag bad'_1 is set then simulation fails with output $(0, \perp)$.
   c. If the flag alert' is set then pick $\mathbf{t}_n \xleftarrow{\$} R_q^k$. Otherwise using a predefined public key $\mathbf{t}$ define $\mathbf{t}_n := \mathbf{t} - \sum_{j=1}^{n-1} \mathbf{t}_j$.
      - If $\mathrm{HT}_2[\mathbf{t}_n, n]$ has been already set then set bad'_2 and simulation fails with output $(0, \perp)$.
      - Otherwise program the random oracle $\mathrm{HT}_2[\mathbf{t}_n, n] := g'_n$ and send out $\mathbf{t}_n$.
3. Upon receiving $\mathbf{t}_j$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_2(\mathbf{t}_j, j) \neq g'_j$ for some j then send out $\mathtt{abort}$.
   b. If alert' is set and $\mathsf{H}_2(\mathbf{t}_j, j) = g'_j$ for all j then set bad'_3 and simulation fails with output $(0, \perp)$.

If neither the protocol aborts nor the simulation fails, the simulator obtains public key share $\mathbf{t}_n$ and $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ as local output.

Figure 4.12: Key generation simulator for DS_2

Algorithm $\mathsf{SimSign}_n(\mathsf{sid}, \mathbf{t}_n, \mathsf{pk}, \mu)$

The simulator is parameterized by public parameters described in Table 4.2 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{\mathsf{ck}}$. The simulator assumes that $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \mathbf{t})$ (Fig. 4.12) has been previously invoked. If the simulator halts with `abort` at any point, then all $\mathsf{SimSign}_n(\mathsf{sid}, \mathbf{t}_n, \mathsf{pk}, \mu)$ executions are aborted.

Inputs

1. The simulator receives a unique session ID sid, $\mathbf{t}_n$, $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and message $\mu \in \{0,1\}^*$ as input.
2. The simulator verifies that sid has not been used before (if it has been, the protocol is not executed).
3. The simulator locally computes a per-message commitment key by querying a random oracle $\mathsf{tck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$. If $\mathrm{TDT}[\mu, \mathsf{pk}] = \perp$ (i.e., TCGen was not called) then set bad_4 and simulation fails with output $(0, \perp)$. Otherwise obtain the trapdoor $\mathsf{td} \leftarrow \mathrm{TDT}[\mu, \mathsf{pk}]$.

Signature Generation

1. Compute the first message as follows.
   a. Compute $\mathsf{com}_n \leftarrow \mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$.
   b. Send out com_n.
2. Upon receiving com_j for all $j \in [n-1]$ compute the signature share as follows.
   a. Set $\mathsf{com} := \sum_{j\in[n]} \mathsf{com}_j$.
   b. Derive a challenge $c \leftarrow \mathsf{H}_0(\mathsf{com}, \mu, \mathsf{pk})$.
   c. Computes a simulated signature share $\mathbf{z}_n \xleftarrow{\$} D_s^{\ell+k}$ and derive randomness $r_n \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_n, \mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$.
   d. With probability $1/M$ send out $(\mathbf{z}_n, r_n)$; otherwise send out `restart` and go to 1.
3. Upon receiving `restart` from some party go to 1. Otherwise upon receiving $(\mathbf{z}_j, r_j)$ for all $j \in [n-1]$ compute the combined signature as follows
   a. For each $j \in [n-1]$ reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ and validate the signature share:
   $$\|\mathbf{z}_j\|_2 \le B \text{ and } \mathsf{Open}_{\mathsf{tck}}(\mathsf{com}_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out `abort`.
   b. Compute $\mathbf{z} := \sum_{j\in[n]} \mathbf{z}_j$ and $r := \sum_{j\in[n]} r_j$.

If neither the protocol aborts nor the simulation fails, the simulator obtains a signature $(\mathsf{com}, \mathbf{z}, r)$ as local output.

Figure 4.13: Signature generation simulator for DS_2.

4.4 MS_2: Two-round Multi-signature in the Plain Public Key Model

In this section we describe our two-round multi-signature scheme MS_2. The main difference from n-out-of-n signature is that, the protocol requires no interactive key generation at

Protocol 3: $\mathsf{MS}_2.\mathsf{Sign}(\mathsf{sid}, \mathsf{sk}_n, \mathsf{pk}_n, \mu, L)$

The protocol is parameterized by public parameters described in Table 4.2 and matrix $\bar{\mathbf{A}}$, and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{\mathsf{ck}}$. The protocol assumes that $\mathsf{MS}_2.\mathsf{Gen}(\mathsf{pp})$ has been previously invoked. If a party halts with `abort` at any point, then all $\mathsf{Sign}(\mathsf{sid}, \mathsf{sk}_n, \mathsf{pk}_n, \mu, L)$ executions are aborted.

Inputs

1. P_n receives a unique session ID sid, $\mathsf{sk}_n = \mathbf{s}_n$, $\mathsf{pk} = \mathbf{t}_n$, message $\mu \in \{0,1\}^*$ and a list of public keys L as input. If $n' := |L| > n$ or $\mathbf{t}_n \notin L$ then send out `abort`. Otherwise parse L as $\{\mathbf{t}_1, \ldots, \mathbf{t}_{n'-1}, \mathbf{t}_n\}$.
2. P_n verifies that sid has not been used before (if it has been, the protocol is not executed).
3. P_n locally computes a per-message commitment key $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, L)$.

Signature Generation P_n works as follows:

1. Compute the first message as follows.
   a. Sample $\mathbf{y}_n \xleftarrow{\$} D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   b. Compute $\mathsf{com}_n \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathbf{w}_n; r_n)$ with $r_n \xleftarrow{\$} D(S_r)$.
   c. Send out com_n.
2. Upon receiving com_j for all $j \in [n'-1]$ compute the signature share as follows.
   a. Set $\mathsf{com} := \sum_{j\in[n'-1]} \mathsf{com}_j + \mathsf{com}_n$.
   b. Derive a challenge $c_n \leftarrow \mathsf{H}_0(\mathbf{t}_n, \mathsf{com}, \mu, L)$.
   c. Computes a signature share $\mathbf{z}_n := c_n\mathbf{s}_n + \mathbf{y}_n$.
   d. Run the rejection sampling on input $(c_n\mathbf{s}_n, \mathbf{z}_n)$, i.e., with probability
   $$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c_n\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$
   send out $(\mathbf{z}_n, r_n)$; otherwise send out `restart` and go to 1.
3. Upon receiving `restart` from some party go to 1. Otherwise upon receiving $(\mathbf{z}_j, r_j)$ for all $j \in [n'-1]$ compute the combined signature as follows
   a. For each $j \in [n'-1]$ derive a per-user challenge $c_j \leftarrow \mathsf{H}_0(\mathbf{t}_j, \mathsf{com}, \mu, L)$, reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j\mathbf{t}_j$ and validate the signature share:
   $$\|\mathbf{z}_j\|_2 \le B \text{ and } \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out `abort`.
   b. Compute $\mathbf{z} := \sum_{j\in[n'-1]} \mathbf{z}_j + \mathbf{z}_n$ and $r := \sum_{j\in[n'-1]} r_j + r_n$.

If the protocol does not abort, P_n obtains a signature $(\mathsf{com}, \mathbf{z}, r)$ as local output.

Algorithm $\mathsf{MS}_2.\mathsf{Ver}((\mathsf{com}, \mathbf{z}, r), \mu, L)$

Upon receiving a message μ, signature $(\mathsf{com}, \mathbf{z}, r)$, and a set of public keys L, if $|L| > n$ then reject the signature. Otherwise for each j such that $\mathbf{t}_j \in L$ derive a per-user challenge $c_j \leftarrow \mathsf{H}_0(\mathbf{t}_j, \mathsf{com}, \mu, L)$ and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_j c_j\mathbf{t}_j$. Then accept if $\|\mathbf{z}\|_2 \le B_n$ and $\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, r, \mathbf{w}) = 1$.

Figure 4.14: Two-round multi-signature secure in the plain public key model. The differences with Fig. 4.7 are highlighted in orange.

all, and instead for each signing execution a party receives a set of public keys L together with a message to be signed. As the number of participants may change for each signing attempt, in this section we define n to be the *maximum number of signers allowed in a single execution of signing protocol*, i.e., only L of cardinalty at most n is a valid input.

Now we present concrete specifications of $\mathsf{MS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$. The Setup works just like the one for DS_2, but it additionally outputs a matrix $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$ as part of public parameters, so we assume that $\bar{\mathbf{A}}$ is generated by a trusted third party (if the generation of $\bar{\mathbf{A}}$ has to be done in a distributed way then parties can invoke a matrix generation protocol in Fig. 4.7 instead and a signer only uses $\bar{\mathbf{A}}$ as long as it participated in the generation of $\bar{\mathbf{A}}$). Then Gen algorithm is the same as Algorithm 6, except that it takes $\bar{\mathbf{A}}$ as input and outputs $\mathsf{sk} = \mathbf{s} \in S_\eta^{\ell+k}$ and $\mathsf{pk} = \mathbf{t} \in R_q^k$. In a multi-signature scheme, the indices assigned to the signers are just local references to the cosigners participating in a particular protocol instance [BN06], and therefore we wlog assume that each signer assign the index n to itself, and consider other signers' indices as $1, \ldots, n' - 1$, where $n' = |L| \le n$. The signing protocol Sign and verification Ver are described in Fig. 4.14. The only difference from $\mathsf{DS}_2.\mathsf{Sign}_n$ and $\mathsf{DS}_2.\mathsf{Ver}$ is that signature shares are now constructed from per-user challenges, instead of a single common challenge for all co-signers (just as Bellare–Neven [BN06] or Bagherzandi et al. [BCJ08] did). Therefore, the random oracle simulation below is more involved than in the proof for Theorem 4.1. On the other hand, as MS_2 has no interactive key generation, the proof only requires much simpler key generation simulation. Therefore, the concrete security bound in the following theorem is slightly better than the previous case. Proof is deferred to [DOTT20].

Theorem 4.2. *Suppose the trapdoor commitment scheme* TCOM *is secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that initiates* Q_s *signature generation protocols by querying* $\mathcal{O}_n^{\mathsf{MS}_2}$*, and makes* Q_h *queries to the random oracle* $\mathsf{H}_0, \mathsf{H}_3$*, the protocol* MS_2 *of Fig. 4.14 is* $\mathsf{MS\text{-}UF\text{-}CMA}$ *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions, where* $\beta = 2\sqrt{B_n^2 + \kappa}$*. Concretely, using other parameters specified in Table 4.2, the advantage of* $\mathcal{A}$ *is bounded as follows.*

$$\mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}_2}(\mathcal{A}) \le e \cdot (Q_h + Q_s + 1) \cdot \Bigg((Q_h + Q_s)\epsilon_{td} + Q_s \cdot \frac{2e^{-t^2/2}}{M} + \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}} + \frac{Q_h + Q_s + 1}{|C|} + \sqrt{(Q_h + Q_s + 1) \cdot \left(\epsilon_{\mathsf{bind}} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}\right)} \Bigg)$$

4.5 Lattice-Based Commitments

In this section, we describe possible constructions for the lattice-based commitment schemes used in our protocols. The three-round protocol DS_3 requires a statistically binding, computationally hiding homomorphic commitment scheme, whereas the two-round protocol DS_2 of Section 4.3 needs a statistically hiding *trapdoor* homomorphic commitment scheme. We show that both types of commitments can be obtained using the techniques of Baum et al. [BDL+18]. More precisely, the first type of commitment scheme is a simple variant of the scheme of [BDL+18], in a parameter range that ensures statistical instead of just computational binding. The fact that such a parameter choice is possible is folklore, and does in fact appear in an earlier version of [BDL+18], so we do not claim any novelty in that regard.

Protocol 4: Lattice-based statistically binding commitment

$\mathsf{CSetup}(1^\kappa)$ takes a security parameter and outputs $\mathsf{cpp} = (q, N, k, m, m', \eta)$.

$\mathsf{CGen}(\mathsf{cpp})$ takes a commitment parameter and outputs ck consisting of $\hat{\mathbf{A}}_1 \in R_q^{m \times m'}$ and $\hat{\mathbf{A}}_2 \in R_q^{k \times m'}$.

$$\hat{\mathbf{A}}_1 = [\mathbf{I}_m | \hat{\mathbf{A}}'_1] \text{ where } \hat{\mathbf{A}}'_1 \xleftarrow{\$} R_q^{m \times (m'-m)}$$
$$\hat{\mathbf{A}}_2 = [\mathbf{0}^{k \times m} | \mathbf{I}_k | \hat{\mathbf{A}}'_2] \text{ where } \hat{\mathbf{A}}'_2 \xleftarrow{\$} R_q^{k \times (m'-m-k)}$$

$\mathsf{Commit}_{\mathsf{ck}}(\mathbf{x})$ takes $\mathbf{x} \in R_q^k$, samples the randomness vector $\mathbf{r} \xleftarrow{\$} D_s^{m'}$ and outputs

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{A}}_1 \\ \hat{\mathbf{A}}_2 \end{bmatrix} \cdot \mathbf{r} + \begin{bmatrix} \mathbf{0}^m \\ \mathbf{x} \end{bmatrix}.$$

$\mathsf{Open}_{\mathsf{ck}}(\mathbf{f}_1, \mathbf{f}_2, \mathbf{x}, \mathbf{r})$ checks that

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{A}}_1 \\ \hat{\mathbf{A}}_2 \end{bmatrix} \cdot \mathbf{r} + \begin{bmatrix} \mathbf{0}^m \\ \mathbf{x} \end{bmatrix} \text{ and } \|\mathbf{r}\|_2 \leq B$$

Figure 4.15: Statistically binding homomorphic commitment from [BDL+18]

The construction of a lattice-based trapdoor commitment scheme does not seem to appear in the literature, but we show that it is again possible by combining [BDL+18] with Micciancio–Peikert style trapdoors [MP12]. To prevent statistical learning attacks on the trapdoor sampling, however, it is important to sample the randomness in the commitment according to a discrete Gaussian distribution, in contrast with Baum et al.'s original scheme.

4.5.1 Statistically Binding Commitment Scheme

We first describe a statistically binding commitment scheme from lattices. The scheme, described in Fig. 4.15, is a simple variant of the scheme from [BDL+18], that mainly differs in the choice of parameter regime: we choose parameters so as to make the underlying SIS problem vacuously hard, and hence the scheme statistically binding. Another minor change is the reliance on discrete Gaussian distributions, for somewhat more standard and compact LWE parameters. The correctness and security properties, as well as the constraints on parameters, are obtained as follows.

Correctness. By construction. We select the bound B as $\Omega(s \cdot \sqrt{m' \cdot N})$. By [MP12, Lemma 2.9], this ensures that the probability to retry in the committing algorithm is negligible.

Statistically binding. Suppose that an adversary can construct a commitment $\mathbf{f}$ on two distinct messages $x \neq x'$, with the associated randomness $\mathbf{r}, \mathbf{r}'$. Since $x \neq x'$, the correctness condition ensures that $\mathbf{r}$ and $\mathbf{r}'$ are distinct and of norm $\leq B$, and satisfy $\hat{\mathbf{A}}_1 \cdot (\mathbf{r} - \mathbf{r}') \equiv 0 \pmod{q}$. This means in particular that there are non zero elements in the Euclidean ball $B_{m'}(0, 2B)$ of radius $2B$ in $R_q^{m'}$ that map to $\mathbf{0}$ in R_q^m. But this happens

with negligible probability on the choice of $\hat{\mathbf{A}}_1$ when $|B_{m'}(0,2B)|/q^{mN} = 2^{-\Omega(N)}$. Now $|B_{m'}(0,2B)| = o((2\pi e/m'N)^{m'N/2} \cdot (2B)^{m'N})$. Hence, picking for example $m' = 2m$, we get:

$$\frac{|B_{m'}(0,2B)|}{q^{mN}} \ll \Big(\frac{4\pi e \cdot B^2}{mNq}\Big)^{mN},$$

and the condition is satisfied for example with $q > 8\pi e B^2/mN$.

Computationally hiding. The randomness $\mathbf{r}$ can be written in the form $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{s}]^T$ where $\mathbf{r}_1 \in R_q^m$, $\mathbf{r}_2 \in R_q^k$, $\mathbf{s} \in R_q^{m'-m-k}$ are all sampled from discrete Gaussians of parameter s.

The commitment elements then become:

$$\mathbf{f}_1 = \mathbf{r}_1 + \hat{\mathbf{A}}'_1 \cdot \begin{bmatrix}\mathbf{r}_2\\ \mathbf{s}\end{bmatrix} \qquad\qquad \mathbf{f}_2 = \mathbf{r}_2 + \hat{\mathbf{A}}'_2 \cdot \mathbf{s} + \mathbf{x},$$

and distinguishing those values from uniform are clearly instances of decision MLWE. Picking $k = m$, $m' = 2m$, $s = \Theta(\sqrt{mN})$, $B = \Theta(mN)$, $q = \Theta((mN)^{3/2})$ yields a simple instatiation with essentially standard security parameters.

4.5.2 Trapdoor Commitment Scheme

We now turn to the construction of a trapdoor commitment scheme with suitable homomorphic properties for our purposes. Our proposed scheme is described in Fig. 4.16. It is presented as a commitment for a *single* ring element $x \in R_q$. It is straightforward to extend it to support a vector $\mathbf{x} \in R_q^k$, but the efficiency gain from doing so is limited compared to simply committing to each coefficient separately, so we omit the extension.

We briefly discuss the various correctness and security properties of the scheme, together with the constraints that the various parameters need to satisfy. In short, we need to pick the standard deviation of the coefficients of the trapdoor matrix $\mathbf{R}$ large enough to ensure that the trapdoor key is statistically close to a normal commitment key; then, the randomness $\mathbf{r}$ in commitments should have large enough standard deviation to make commitments statistically close to uniform (and in particular statistically hiding), and also be sampleable using the trapdoor. These are constraints on $\bar{s}$ and s respectively. Finally, the bound B for verification should be large enough to accomodate valid commitments, and small enough compared to q to still make the scheme computationally binding (which corresponds to the hardness of an underlying Ring-SIS problem). Let us now discuss the properties one by one.

Correctness. By construction. We select the bound B as $C \cdot s \cdot \sqrt{N}(\sqrt{\ell + 2w} + 1)$ where $C \approx 1/\sqrt{2\pi}$ is the constant in [MP12, Lemma 2.9]. By that lemma, this ensures that the probability to retry in the committing algorithm is negligible (and in particular, the distribution of $\mathbf{r}$ after the rejection sampling is statistically close to the original Gaussian).

Computationally binding. Suppose that an adversary can construct a commitment $\mathbf{f}$ on two distinct messages $x \neq x'$, with the associated randomness $\mathbf{r}, \mathbf{r}'$. Since $x \neq x'$, the correctness condition ensures that $\mathbf{r}$ and $\mathbf{r}'$ are distinct and of norm $\leq B$, and satisfy $\hat{\mathbf{A}}_1 \cdot (\mathbf{r} - \mathbf{r}') \equiv 0 \pmod q$ where $\hat{\mathbf{A}}_1$ is the first row of $\hat{\mathbf{A}}$. Therefore, the vector $\mathbf{z} = \mathbf{r} - \mathbf{r}'$ is a solution of the Ring-SIS problem with bound $2B$ associated with $\hat{\mathbf{A}}_1$ (or equivalently, to the $\mathsf{MSIS}_{q,1,\ell+2w-1,2B}$ problem), and finding such a solution is hard.

Note that since the first entry of $\hat{\mathbf{A}}_1$ is invertible, one can put it in the form $[\mathbf{A}|\mathbf{I}]$ without loss of generality to express it directly as an MSIS problem in the sense of Definition 4.2. It

also reduces tightly to standard Ring-SIS, because a random row vector in $R_q^{\ell+2w}$ contains an invertible entry except with probability at most $(N/q)^{\ell+2w} = 1/N^{\Omega(\log N)}$, which is negligible.

Statistically hiding. It suffices to make sure that

$$\hat{\mathbf{A}} \cdot D_s^{\ell+2w} \approx_s U(R_q^2)$$

with high probability on the choice of $\hat{\mathbf{A}}$. This is addressed by [LPR13, Corollary 7.5], which shows that it suffices to pick $s > 2N \cdot q^{(2+2/N)/(\ell+2w)}$.

Indistinguishability of the trapdoor. To ensure that the commitment key $\hat{\mathbf{A}}$ generated by TCGen is indistinguishable from a regular commitment key, it suffices to ensure that $\bar{\mathbf{A}}\mathbf{R}$ is statistically close to uniform. Again by [LPR13, Corollary 7.5], this is guaranteed for $\bar{s} > 2N \cdot q^{(2+2/N)/\ell}$. By setting $\ell = w = \lceil \log_2 q \rceil$, we can thus pick $\bar{s} = \Theta(N)$.

Equivocability. It is clear that an $\mathbf{r}$ sampled according to the given lattice coset discrete Gaussian is distributed as in the regular commitment algorithm (up to the negligible statistical distance due to rejection sampling). The only constraint is thus on the Gaussian parameter that can be achieved by the trapdoor Gaussian sampling algorithm. By [MP12, §5.4], the constraint on s is as follows:

$$s \geq \|\mathbf{R}\| \cdot \omega(\sqrt{\log N})$$

where $\|\mathbf{R}\| \leq C \cdot \bar{s}\sqrt{N}(\sqrt{\ell} + \sqrt{2w} + 1)$ by [MP12, Lemma 2.9]. Thus, one can pick $s = \Theta(N^{3/2} \log^2 N)$.

From the previous paragraphs, we can in particular see that the trapdoor commitment satisfies the security requirements of Definition 4.4. Thus, to summarize, we have proved the following theorem.

Theorem 4.3. *The trapdoor commitment scheme of Fig. 4.16, with the following choice of parameters:*

$$\begin{array}{lll} \bar{s} = \Theta(N) & s = \Theta(N^{3/2} \log^2 N) & B = \Theta(N^2 \log^3 N) \\ \ell = w = \lceil \log_2 q \rceil & q = N^{2+\varepsilon} & (\varepsilon > 0,\ q \text{ prime}). \end{array}$$

is a secure trapdoor commitment scheme assuming that the $\mathsf{MSIS}_{q,1,\ell+2w-1,2B}$ *problem is hard.*

Note that we did not strive for optimality in the parameter selection; a finer analysis is likely to lead to a more compact scheme.

Furthermore, although the commitment has a linear structure that gives it homomorphic features, we need to increase parameters slightly to support additive homomorphism: this is because the standard deviation of the sum of n randomness vectors $\mathbf{v}$ is $\sqrt{n}$ times larger. Therefore, B (and accordingly q) should be increased by a factor of $\sqrt{n}$ to accomodate for n-party additive homomorphism. For constant n, of course, this does not affect the asymptotic efficiency.

Protocol 5: Lattice-based Commitment Scheme

$\mathsf{CSetup}(1^\kappa)$ takes a security parameter and outputs $\mathsf{cpp} = (N, q, \bar{s}, s, B, \ell, w)$.

$\mathsf{CGen}(\mathsf{cpp})$ takes a commitment parameter and samples $\hat{a}_{1,1} \xleftarrow{\$} R_q^\times$ (a uniform invertible element of R_q) and $\hat{a}_{1,j} \xleftarrow{\$} R_q$ for $j = 2, \ldots, \ell + 2w$, $\hat{a}_{2,j} \xleftarrow{\$} R_q$ for $j = 3, \ldots, \ell + 2w$. It then outputs:

$$\hat{\mathbf{A}} = \begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \hat{a}_{1,3} & \cdots & \hat{a}_{1,\ell+2w} \\ 0 & 1 & \hat{a}_{2,3} & \cdots & \hat{a}_{2,\ell+2w} \end{bmatrix}$$

as ck.

$\mathsf{Commit}_{\mathsf{ck}}(x)$ takes $x \in R_q$ and samples a discrete Gaussian vector of randomness $\mathbf{r} \xleftarrow{\$} D_s^{\ell+2w}$. It then outputs

$$\mathbf{f} = \hat{\mathbf{A}} \cdot \mathbf{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \in R_q^2.$$

To ensure perfect correctness, retry unless $\|\mathbf{r}\|_2 \leq B$.

$\mathsf{Open}_{\mathsf{ck}}(\mathbf{f}, \mathbf{r}, x)$ takes commitments, randomness and message, and checks that

$$\mathbf{f} = \hat{\mathbf{A}} \cdot \mathbf{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \text{ and } \|\mathbf{r}\|_2 \leq B.$$

$\mathsf{TCGen}(\mathsf{cpp})$ takes a commitment parameter and samples $\bar{\mathbf{A}} \in R_q^{2\times\ell}$ of the form:

$$\bar{\mathbf{A}} = \begin{bmatrix} \bar{a}_{1,1} & \bar{a}_{1,2} & \bar{a}_{1,3} & \cdots & \bar{a}_{1,\ell} \\ 0 & 1 & \bar{a}_{2,3} & \cdots & \bar{a}_{2,\ell} \end{bmatrix}$$

where all the $\bar{a}_{i,j}$ are uniform in R_q, except $\bar{a}_{1,1}$ which is uniform in $R_q^\times$. It also samples $\mathbf{R} \xleftarrow{\$} D_{\bar{s}}^{\ell\times 2w}$ with discrete Gaussian entries. It then outputs $\mathbf{R}$ as the trapdoor td and $\hat{\mathbf{A}} = [\bar{\mathbf{A}} | \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ as the commitment key tck, where $\mathbf{G}$ is given by:

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & \cdots & 2^{w-1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 2 & \cdots & 2^{w-1} \end{bmatrix} \in R^{2\times 2w}.$$

$\mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$ simply returns a uniformly random commitment $\mathbf{f} \xleftarrow{\$} R_q^{2\times 1}$. There is no need to keep a state.

$\mathsf{Eqv}_{\mathsf{tck}}(\mathbf{R}, \mathbf{f}, x)$ uses the trapdoor discrete Gaussian sampling algorithm of Micciancio–Peikert [MP12, Algorithm 3] (or faster variants such as the one described in [GM18]) to sample $\mathbf{r} \xleftarrow{\$} D_{\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{A}}),s}$ according to the discrete Gaussian of parameter s supported on the lattice coset:

$$\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{A}}) = \{\mathbf{z} \in R^{\ell+2w} \ : \ \hat{\mathbf{A}} \cdot \mathbf{z} \equiv \mathbf{u} \pmod q\} \quad \text{where} \quad \mathbf{u} = \mathbf{f} - \begin{bmatrix} 0 \\ x \end{bmatrix}.$$

Figure 4.16: Equivocable variant of the commitment from [BDL+18].

Chapter 5

Verifiable Encryption from MPC-in-the-Head

5.1 Introduction

A verifiable encryption (VE) scheme is a public-key encryption scheme where one party (called a *prover* $\mathcal{P}$) can encrypt data w with a public key pk (of which the corresponding decryption key sk is held by the *receiver* $\mathcal{R}$), and convince a third party (called the *verifier* $\mathcal{V}$) that the data satisfies some relation R, i.e., $R(x, w) = 1$ with respect to some public statement x. At a very high-level, an (interactive) VE scheme should satisfy the following security properties [CD00]:

- **Completeness**: If $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{R}$ are honest then $\mathcal{V}$ accepts after interacting with $\mathcal{P}$, and $\mathcal{R}$ uses sk to obtain a plaintext w satisfying $R(x, w) = 1$.
- **Zero knowledge**: As $\mathcal{V}$ does not have the decryption key sk, she learns nothing about the plaintext from interacting with $\mathcal{P}$.
- **Validity**: If $\mathcal{V}$ accepts after interacting with a prover $\mathcal{P}^*$, $\mathcal{R}$ is guaranteed to obtain a plaintext w such that $R(x, w) = 1$, even if $\mathcal{P}^*$ is malicious.

Our motivating example for verifiable encryption is the *verifiable backup problem*, where a cryptographic device (such as a hardware security module (HSM)) or cloud service (such as [AWS22a, AWS22b, AKV22, GK22]) that is entrusted to store key material must securely export it for backup in case of hardware failure. These backups must be encrypted (or "wrapped") with the public key of another device, so that the plaintext keys are never exposed outside of the secure hardware [YC22, PK15]. The administrator of the device, responsible for creating backups, does not get assurance that the backup is well-formed, and will import successfully on the new device. She could try the import operation, but this may be expensive (e.g., if the backup device is in a separate facility), or risky (as it spreads the key around more than necessary). This latter risk is well illustrated in the case of cloud-based HSMs, where testing a backup by importing a key into a secondary cloud provider greatly expands the trust boundary.

Even if the import operation succeeds, the admin should still test that the imported private key corresponds to the expected public key, which typically requires using it to create a test signature or decryption. This is undesirable for two reasons: it adds extra use(s) of the key which must be logged for auditing, and it may also involve using the key

for a different purpose than it was created for. Ideally, the exporting device could prove to the administrator that the ciphertext is a well-formed encryption under the receiving device's public key, and further, that the plaintext is a private key corresponding to a particular public key, e.g., the device claims "I encrypted the ECDSA signing key x for a public verification key y" and the administrator should be convinced that $y = g^x$ without access to the plaintext x. If the exported key is a symmetric key, then the device should prove that the plaintext is a key consistent with a commitment to the key, or a ciphertext or MAC created with the key. Verifiable encryption is a natural solution to this problem.

Verifiable Encryption Despite being introduced more than two decades ago by Stadler [Sta96] and becoming a well-defined primitive with a relatively general solution in the work of Camenisch and Damgård [CD00], constructions suitable for the verifiable backup problem are limited. There are multiple challenges. We need generality, to allow multiple types of relation to be supported, not only a single one (as in [CS03, NRSW20, LN17]). Our use case requires verifiable encryption of many types of keys (potentially all the types here [PK22]), and at least ECC, RSA, and AES (the common types supported by cloud providers [AWS22a, AKV22, GK22]). We also want to minimize the additional assumptions required, ideally not requiring any new assumptions; for example if an AES key is to be exported, encrypted under an RSA key, we should not need to make assumptions in elliptic curve groups (perhaps with a pairing), as might be the case if certain SNARK proof systems were used for verifiability [Gro16, MBKM19, BBB+18, LCKO19]. We also want flexibility in the receiver's public-key encryption (PKE) scheme, again to minimize new assumptions, but also to support security goals like threshold decryption or post-quantum security, rather than have a VE scheme that dictates the PKE the receiver must use (as in [CS03, NRSW20, LN17]). Finally, performance must be good enough for use in practice, which excludes using fully general proof systems (e.g., [Mic00, GMW87, GOS06]). In summary, we desire a construction that is as general as possible, introduces no new assumptions, and is performant enough to be practical.

There are multiple applications of verifiable encryption in the literature. Some early examples are publicly verifiable secret sharing [Sta96], and verifiable encryption of signatures for optimistic fair exchange [ASW98, Ate99]. Key escrow [YY98, PS00b], where parties encrypt their private key to a trusted escrow authority, can be achieved with verifiable encryption, since it becomes possible for other parties on the network to ensure that the correct key has been escrowed. A common theme is *identity escrow* (or revokable anonymity) in privacy systems and group signatures, where an anonymous party encrypts their identity for an authority, who can de-anonymize them under certain circumstances. In cryptographic voting systems, voters often encrypt their votes and prove that their selection is in a set of valid choices (e.g., in $\{0, 1\}$ to encode a "yes" or "no" vote). The earliest paper with this idea predates the literature on verifiable encryption [CF85] and VE is still used in cryptographic voting systems today, see for example [EG21, CCFG16].

ZK from MPC The MPC-in-the-head (MPCitH) paradigm [IKOS07] is a way to create a zero-knowledge (ZK) proof for a relation R, given a secure multiparty computation protocol (MPC) to compute R. Some of the advantages of this approach make it well suited to our verifiable encryption problem. First, MPC protocols are *very flexible*, so that we can instantiate ZK proofs for many choices of R, typically expressed as binary or arithmetic circuits. The paradigm extends beyond circuits as well: we give an MPCitH protocol to prove knowledge of a discrete logarithm, and use our results to verifiably encrypt discrete logs.

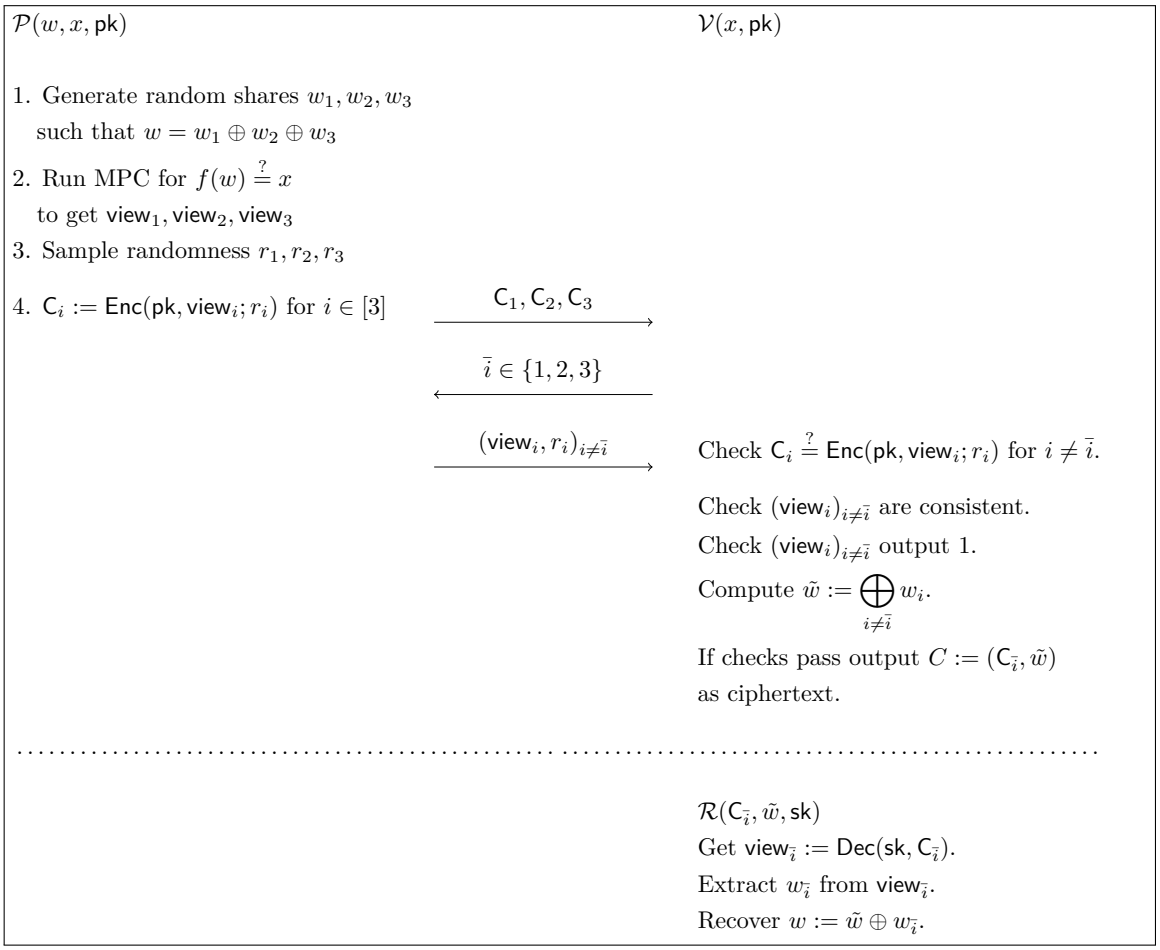

Figure 5.1: High-level overview of our transform, applied to ZKBoo.

Second, if the MPC protocol is information theoretically secure, converting it to a ZK proof only requires a secure commitment scheme, which can be instantiated with a cryptographic hash function, so that the proof system requires minimal assumptions, and is post-quantum secure. Finally, the performance of MPCitH proof systems in terms of prover and verification costs and proof sizes are practical, and have been steadily improving as has been demonstrated in the area of post-quantum signatures. To use the AES-128 circuit as an example, proof sizes went from 209 KB [GCZ16] to 32 KB [dDOS19] to 13 KB [BdK+21a] in the past five years, and the running time of the prover and verifier is roughly 50ms (see the implementation benchmarks in [BdK+21a]). Taken together, these properties will allow us to construct verifiable encryption schemes that are very general, make minimal assumptions, achieve post-quantum (PQ) security and are efficient enough for practical use.

5.1.1 Our Contributions and Techniques

Our results apply to a broad class of MPCitH proofs: those that can be viewed as an interactive oracle proof (IOP). The original class from [IKOS07] is captured by the IOP framework as well as many more recent MPCitH proofs aimed at concrete efficiency, such as [GMO16, KKW18, BdK+21a, BN20, dOT21, Bd20, Beu20]. The IOP framework

of [BCS16] allows a modular design, and comes with definitions of zero-knowledge and straight-line extractability that we need to prove our results. Revisiting some existing MPCitH proofs as IOPs is a side contribution of this paper.

Generic compiler for MPC-in-the-head-based VE In Section 5.3 we give a compiler that takes a proof protocol from the MPCitH-IOP class and converts it into a verifiable encryption scheme, denoted MPCitH-VE. We describe MPCitH-VE as a public-coin three-round interactive protocol, which can be made non-interactive using the standard Fiat-Shamir transform [FS87]. An abstract protocol MPCitH-IOP captures several three-round protocols, including [IKOS07], ZKBoo [GMO16], ZKB++ [CDG+17], and our new DKG-in-the-head protocol described below. We also discuss using essentially the same idea to compile KKW-IOP and Banquet-IOP (IOP versions of [KKW18] and [BdK+21a], respectively).

The other input to the compiler is a public key encryption (PKE) scheme, such as Elgamal, RSA-OAEP or PQ-secure options like Kyber [SAB+20] or FrodoKEM [NAB+19]. We define and prove the requirements the PKE must have to ensure MPCitH-VE is secure. In short, ciphertexts created by the PKE must be a secure commitment (both hiding and binding) to the plaintext. Hiding is provided by CPA security (security against chosen-plaintext attacks), and for binding, we define a new property called *undeniability*, which is trivial for PKE schemes with perfect correctness, but may be absent otherwise. Notably, lattice-based PQ schemes are usually not perfectly correct. In [TZ21] we prove that the Fujisaki-Okamoto transform [FO99, FO13, HHK17] (and simpler variants of it) can be used to upgrade any statistically correct PKE schemes to obtain undeniability, making our construction compatible with many existing schemes. An implication is that encryption schemes using the FO transform are secure commitment schemes, which might be of independent interest.

Our framework is versatile: because the circuit proven by the MPC-in-the-head prover is decoupled from a complex encryption function, the prover's work can be focused on proving properties (i.e., the relation R) about the encrypted data, not on the proof of plaintext knowledge. Proof of plaintext knowledge is achieved with existing mechanisms in the MPCitH proof. Hence, with our approach we can easily instantiate VE with various combinations of R and PKE.

To illustrate the core idea of our transform, we sketch an example VE scheme based on the ZKBoo proof system [GMO16], described at a high-level in Fig. 5.1. The original ZKBoo protocol for relation $R(x, w) := (f(w) \stackrel{?}{=} x)$ (where f is typically some one-way function) proceeds as follows: the prover $\mathcal{P}$ first distributes to three parties additive shares (w_1, w_2, w_3) of the secret witness w. Then $\mathcal{P}$ runs an MPC protocol computing $f(w) \stackrel{?}{=} x$ "in the head", to produce the *view* of each party, i.e., a string consisting of the input share, output, communication, and random tape. $\mathcal{P}$ sends commitments to the views as its first message, and the verifier $\mathcal{V}$ returns a challenge $\bar{i} \in \{1, 2, 3\}$, indicating party $\bar{i}$'s view is supposed to remain secret. $\mathcal{P}$ then responds with the views of party $i \neq \bar{i}$ and commitment randomness. $\mathcal{V}$ accepts if the two commitments are correctly opened and they constitute a correct run of MPC.

Now notice that one can immediately recover the witness once the remaining commitment to party $\bar{i}$ is revealed.[1] Our main observation is that a technique similar to *straight-line extractable* ZK proofs (see Section 5.1.2) gives rise to a secure VE scheme: by replacing commitments in the original proof system with public-key encryptions, the prover

[1] In fact, this is how a knowledge extractor works when proving *knowledge soundness* of the scheme.

$\mathcal{P}$ now sends three ciphertexts containing witness shares: $\mathsf{C}_i := \mathsf{Enc}(\mathsf{pk}, \mathsf{view}_i)$ for $i = 1, 2, 3$. The verifier still learns nothing about the encrypted data w since one of its additive shares is kept encrypted. By contrast, the receiver $\mathcal{R}$ with knowledge of the decryption key sk can decrypt the unopened ciphertext $\mathsf{C}_{\bar{i}}$ (or commitment) to obtain the remaining share $w_{\bar{i}}$, from which the plaintext w can be recovered using the shares $(w_i)_{i \neq \bar{i}}$ revealed in the public transcript. As usual, by applying the Fiat–Shamir transform [FS87], the above interactive protocol can be turned into a non-interactive VE scheme in the random oracle model, as we formally discuss in [TZ21].

Methods for compressing ciphertext In our compiler, essentially the *transcript* itself is output as a ciphertext, after the prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact in MPCitH-VE. While the size of transcript is proportional to the number of parallel repetitions τ required to guarantee negligible soundness error, the receiver $\mathcal{R}$ only needs one of them in case the prover behaves honestly. To close this gap, in Section 5.4 we give two methods to compress the VE ciphertexts. The first, called the *random subset method*, is very simple, incurs no computational overhead, and can reduce ciphertext size by a factor of three when τ is large. If τ is already small, it is also possible to trade ciphertext size for τ, which might be desirable depending on the application.

The second approach, called the *equality proof method*, is optimal as it achieves constant size ciphertexts, $O(|w|)$ (provided PKE has constant ciphertext expansion). However, it requires special properties of PKE, and significantly increases proof size, prover and verifier computational costs, so it is more of a possibility result than a practical construction. We highlight improving compression as an interesting direction for future work.

Concrete instantiations In Section 5.5 we describe concrete approaches to verifiably encrypting discrete logarithms in any prime order group and AES keys.

The former is realized by our new non-interactive ZK protocol for the relation $R = \left\{(y, x) \ : \ y = g^x\right\}$, called *distributed key generation in the head (DKG-in-the-head)*. In this protocol, the prover emulates a protocol where parties run a DKG protocol to compute $y = g^x$. Since the DKG protocol only needs to have passive security and a broadcast channel is available for free in the MPC-in-the-head setting, our proposed protocol is extremely simple, requiring only a single round of interaction between parties.

To verifiably encrypt AES keys, our protocol is derived from the underlying interactive ZK protocol of Banquet [BdK+21a], where $\mathcal{P}$ proves knowledge of an AES key used to generate an AES ciphertext from a public plaintext, i.e., it is specialized for the relation $R = \left\{((\mathsf{ct}, \mathsf{pt}), K) \ : \ \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\right\}$. Prior to this work, there has been no VE scheme for verifiably encrypting AES keys, which may find interesting applications in the post-quantum setting when instantiated with quantum-resilient PKE.

Revisiting the Camenisch-Damgård VE construction As a separate contribution, we revisit the existing verifiable encryption of Camenisch and Damgård [CD00] in [TZ21] and show that it fails to retain the validity property when instantiated with IND-CPA PKE schemes that are only *statistically correct*, as opposed to perfectly correct. We describe concrete attacks in which a malicious prover can convince the verifier to accept a ciphertext that decrypts to random data unrelated to the R. Finally, we show that by additionally assuming the undeniability property their construction can also be securely instantiated with statistically correct PKE schemes.

5.1.2 Related Work

Camenisch–Damgård transform Although our generic transform is similar in spirit to that of [CD00], there are some differences. Our starting point is any MPC-in-the-head IOP with the straight-line extractable property, while [CD00] is focused on 2-special sound Σ-protocols with 1-bit challenge space (though it seems possible to generalize their transform to k-special sound protocols for any k as well). Although one can naïvely apply [CD00] to some MPC-in-the-head protocols with k-special soundness, such as ZKBoo and IKOS, our method directly modifies the committing function and thus leads to better communication complexity. Moreover, [CD00] does not apply to more efficient MPC-in-the-head constructions, including KKW and Banquet: because the challenge spaces of these protocols are not limited to party indices the notion of special soundness is not well-defined. In contrast, our transform relies on straight-line extractability, and therefore applies to KKW and Banquet as well.

Camenisch–Shoup scheme Camenisch and Shoup [CS03] propose protocols for efficient verifiable encryption and decryption of discrete logarithms. However, it only works for discrete logarithms in a group where Paillier's decision composite residuosity (DCR) assumption holds, and the PKE is fixed to (a variant of) Paillier's scheme as well. The scheme is therefore not suitable for encrypting ECC, RSA or AES keys, one of our motivating examples. In theory it is possible to prove that the plaintext of a Camenisch-Shoup ciphertext corresponds to a discrete log from a prime order group. However, this would require range proofs across the two groups, and security still relies on DCR (and possibly more, depending on how the range proofs are done).

SNARK-based constructions Lee et al. [LCKO19] gives a construction of a verifiable encryption scheme that is tailored to use in voting schemes as it is additively homomorphic and supports rerandomization. The construction is pairing-based, Elgamal-like and thus integrates well with SNARK proof systems. Just like our framework, theirs also decouples the encryption function from the circuit describing the relation, using the *commit-and-prove* SNARK of [CFQ19]. It requires a trusted setup assumption due to the use of CRS-based SNARK, while ours is naturally transparent thanks to the underlying MPC-in-the-head paradigm.

Nick et al. [NRSW20] gives a construction which can encrypt a discrete logarithm in an elliptic curve group, using a special PRF called Purify. The scheme does allow, e.g., encryption of an ECDSA private key without any trusted setup assumption thanks to the use of Bulletproofs [BBB+18], but requires that encryption be done with an Elgamal-like PKE. As we compare in Table 5.1, their ciphertext and proof are more compact than those of our DKG-in-the-head VE scheme, while ours requires less prover time. A complication related to implementation of the Purify PRF is that one must choose an additional pair of elliptic curves, related to the group order of the curve where the discrete logarithm is defined, such that the DDH assumption holds. In contrast, our framework does not introduce any additional assumption other than IND-CPA and undeniability of PKE (already satisfied by perfectly correct schemes and many statistical ones as we analyze).

Lattice-based constructions Lyubashevsky and Neven [LN17] give a verifiable encryption scheme for lattices, based on the hardness of the ring learning with errors (RLWE) problem. They give a proof of plaintext knowledge, secure in the ROM that does not use parallel repetition to boost soundness. Their scheme can be further extended to support CCA security. The analysis of our VE construction does not consider CCA security

and it is not "one-shot" as MPC-in-the-head proofs usually rely on parallel repetition or cut-and-choose unlike [LN17]. The construction comes with multiple caveats.

- A malicious prover may create a ciphertext that takes variable time to decrypt. In particular decryption requires $O(q)$ time to decrypt, where q is the number of hash queries made by the prover. This makes decryption potentially very expensive.
- Decryption is not guaranteed to recover the original plaintext, but vectors with small coefficients. It's argued that this is sufficient for some of the applications considered in [LN17], but may not be sufficient in general.
- The size of the proof and ciphertext are relatively large, for example proof sizes are 38–54 KB and ciphertext sizes are 48–71 KB. Proofs and ciphertexts may be as short 9 KB however, this is for verifiable encryption when there is no plaintext.

Isogeny-based construction Beullens et al. [BDK+21b] recently proposed a VE scheme based on isogenies (or more generally, any cryptographically-hard group action). Their main motivation is to construct a building block of a ring signature: their VE prover essentially proves that (1) it encrypted a verification key vk, (2) vk belongs to a ring, and (3) it knows the secret sk corresponding to vk. Although fairly efficient, their approach inherently relies on Elgamal-like PKEs and it is highly specialized for the above limited class of relation. On the other hand, our focus is to build a general framework to support a large class of PKEs and relations as required for concrete solutions to the verifiable key export problem described earlier.

MPCitH and IOP proof systems We briefly survey some of the many existing MPCitH-based proof systems, optimized for different relations, as these immediately give verifiable encryption schemes by applying our transform. [Beu20] gives an MPCitH-based proof of a solution of an SIS (short integer solution) instance. We can apply our transform to construct a verifiable encryption of SIS witnesses (here the witness is exact, not relaxed as in [LN17]). The proof protocols in [Beu20] for other relations, such as the PKP and MQ problems, are also compatible with our transform. [BN20] also gives multiple MPCitH-based proofs for lattice problems (SIS), which are also amenable to our transform, but are outperformed by the proofs of [Beu20]. The Limbo proof system [dOT21] is efficient for general R described as circuits, making it a good choice for hash functions, or as an alternative to Banquet. Gjøsteen et al. [GHM+21] present verifiable *decryption* protocols from MPCitH proofs, by designing suitable distributed decryption protocols for Elgamal and BGV lattice-based encryption schemes.

Aurora [BCR+19] and Ligero [AHIV17] are non-interactive proof systems for R1CS that are constructed by defining an IOP, then making it non-interactive using the transform in [BCS16]. Both have short proofs for relations involving lattices, and Aurora has the shortest proofs for SIS, about 10x shorter than [Beu20, BN20]. As we mention in Section 5.6, for this reason a more general transform for building VE schemes from IOPs in the [BCS16] framework is interesting future work.

Connection between straight-line extraction and verifiable encryption *Straight-line extractability (SLE)* (or sometimes called *online extractability*) is a special type of extractability, specialized to proof systems in the ROM or in the CRS model. The prover commits to witness-dependent strings via extractable commitments instantiated with the RO or PKE, and the extractor is given the statement, the transcript, and the prover's query history (in the ROM) or a secret trapdoor (in the CRS model) to extract a witness. In particular the straight-line extractor does not get any access to the prover, or ability

to rewind them. SLE is especially crucial for security in the QROM, since rewinding techniques are generally prohibitively expensive in that setting. Numerous works achieve SLE of commit-and-open-type proof systems (including MPC-in-the-head) [Pas03, KKW18, DFMS21, HLR21], lattice-based ZK proof systems [Kat21], and straight-line extractable alternatives to the Fiat-Shamir transform [Fis05, Unr15]. A receiver $\mathcal{R}$ of our MPCitH-VE essentially behaves like a straight-line extractor for the MPC-in-the-head proof systems whose commitments are replaced with PKE. In this work, we formally draw a connection between the validity property of VE and SLE of IOP, a setting where commitments are idealized and thus SLE holds very naturally.

Encryption as a Commitment Most natural public key encryption schemes are committing, and constructing a non-committing one (a deniable scheme) is challenging.

[GH03] defines committing public-key encryption, but defines the verification algorithm in a more generic way than what is used in our verifiable encryption scheme and the one of [CD00]. Rather than having the verifier recompute the ciphertext as we do, given the purported (message, randomness) pair, the verify algorithm can be any function that takes as input the message, and a hint produced by the opening function.

[GLR17, DGRW18] looks at committing encryption for symmetric-key AEAD schemes, to support an analysis of a primitive called *message franking*, where participants in a messaging platform can report abusive messages to the service provider. The name *encryptment* is also used, a portmanteau of the terms encryption and commitment. The schemes support many additional features beyond what is required for verifiable encryption in our setting, and the definitions are consequently more complicated than those of [GH03].

[BDD20] recently proved that Pointcheval's IND-CCA PKE [Poi00] can be used as a secure commitment scheme as is, and it is thus plausible that their analysis can be adapted to show undeniability of the scheme as well. Our analysis of the Fujisaki-Okamoto transform also suggests that CCA conversions of this type are useful for obtaining undeniability (and thus binding). It is an interesting follow-up question whether CCA security in general is sufficient for PKE to be committing and/or undeniable.

The opposite of what we need is called *deniable encryption* [CDNO97]. Here the scheme is carefully constructed so that the encryption is *not* a binding commitment to the message and randomness, allowing a sender of a ciphertext to later claim they sent a different message (hence denying the original message). After sending a ciphertext $c = \mathsf{Enc}(m; r)$ then sender can later claim they sent (m', r'), and anyone can check that $c = \mathsf{Enc}(m'; r')$ as well. This is why we use the name *undeniable encryption* to describe a scheme where this is not possible. While a "sender-deniable encryption scheme" in the terminology of [CDNO97] is sufficient as a counterexample to the analysis of [CD00], the example encryption schemes we describe for this purpose in [TZ21] only require a weaker type of deniability.

Non-committing encryption [CFGN96, DN00] is related to deniable encryption, but constructions are interactive and the goal is to improve certain types MPC protocols. Briefly, in the security analysis, the simulator can use the fact that the encryption is not a commitment to be able to create simulated ciphertexts, then later open them to plaintexts that are consistent with later information.

Finally we mention *witness encryption* [GGSW13], which superficially sounds related, since in VE we are encrypting a witness w that is associated to a statement x by a relation R. However, a witness encryption scheme for R is a PKE-like primitive that allows us to use x as a public key and w is the secret key. No witnesses are ever encrypted! Witness encryption can be viewed as an encryption analog to *signatures of knowledge* [CL06], where

w is a signing key and x is a public key verification key.

5.2 Preliminaries

First we introduce some notation and conventions used throughout the paper. The security parameter is denoted κ, and for an integer x, $[x]$ is short for the set $\{1, \ldots, x\}$. Whenever we have a two-part adversary, written as a pair, e.g, $(\mathbf{A}^*, \mathbf{P}^*)$, we assume that $\mathbf{A}^*$ and $\mathbf{P}^*$ share state, and do not explicitly write it as an output of $\mathbf{A}^*$ and an input to $\mathbf{P}^*$. For a set S, we denote by $x \xleftarrow{\$} S$ sampling an element x from S uniformly at random.

In [TZ21], we introduce the standard notions of public-key encryption (PKE), extractable commitments (ECOM), and interactive oracle proofs (IOP).

5.2.1 Extractable commitment from perfectly correct PKE

In the following we show that most commonly used public-key encryption schemes give rise to perfectly binding and computationally hiding commitment schemes. A similar construction appears in [GH03], and is somewhat folklore, below we describe the exact construction we will use, and analyze its security. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. We construct a commitment scheme $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as follows. For simplicity we assume throughout that the message space S_m and random space S_r of the commitment schemes are identical to those of the encryption schemes.

- $\mathsf{CGen}(1^\kappa)$ runs $\mathsf{PKE.Gen}(1^\kappa)$ and outputs pk as the commitment key.
- $\mathsf{Commit}(\mathsf{pk}, m; r)$ outputs $c = \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$.
- The opening of the commitment c is (m, r), and the verifier checks (m, r) against c by computing $c' = \mathsf{Enc}(\mathsf{pk}, m, r)$; the opening is accepted iff $c' = c$, $m \in S_m$ and $r \in S_r$.
- $\mathsf{CExt}(\mathsf{sk}, c)$ outputs $m = \mathsf{PKE.Dec}(\mathsf{sk}, c)$.

We now show this is a secure commitment scheme for perfectly correct, IND-CPA secure encryption schemes. The two most commonly used choices of PKE, RSA and Elgamal, both meet these requirements, and can be used as commitment schemes. Proof is deferred to [TZ21].

Lemma 5.1. *If* PKE *is perfectly correct and* ϵ_{cpa}*-IND-CPA secure, the above commitment scheme is perfectly extractable, perfectly binding and* ϵ_{hide}*-computationally hiding with* $\epsilon_{\mathsf{hide}} \leq \epsilon_{\mathsf{cpa}}$.

Note that for encryption schemes that are not perfectly correct, there can exist (m, m', r, r') such that $\mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$. In the full version [TZ21] we will show two examples of such schemes, one based on decisional composite residuosity, and one based on the learning with errors (LWE) problem. In general, the base encryption scheme of post-quantum lattice-based candidates like FrodoKEM [NAB+19] and Kyber [SAB+20] are CPA secure, but not perfectly correct, and even the complete CCA-secure schemes may still be incorrect with bounded probability.

5.2.2 Verifiable encryption

We define a secure verifiable encryption scheme by adapting the definition from [CD00]. Non-interactive VE is formally defined in [TZ21]. The main difference with [CD00] is that we additionally consider a *compression* algorithm $\mathcal{C}$ that takes a transcript exchanged between a prover and a verifier, and outputs a corresponding ciphertext. In practice, $\mathcal{C}$ would be run by the verifier right after interacting with the prover and obtaining a valid transcript. We explicitly introduce this because our proposed construction will benefit from different optimization strategies that postprocess accepting transcripts to produce a highly compressed ciphertext. Moreover, unlike [CD00] we only consider ZK against *honest* verifiers, since this is sufficient to prove ZK of non-interactive VE in the random oracle model using the Fiat-Shamir transform.

Definition 5.1 (Verifiable Encryption Scheme)**.** *Let R be a relation and $L_R := \{x : \exists w : (x,w) \in R\}$. A secure verifiable encryption scheme VE_R for R consists of a tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{R})$:*

- *$\mathcal{G}(1^\kappa)$: A key generation algorithm that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.*
- *$(\mathcal{P}, \mathcal{V})$: A two-party protocol, where both $\mathcal{P}$ and $\mathcal{V}$ take (x, pk) and $\mathcal{P}$ additionally takes a plaintext w as inputs. We let $(b, \mathsf{tr}) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x)$ denote the output pair of $\mathcal{V}$ on common input (pk, x) when interacting with $\mathcal{P}(w)$, where $b \in \{0,1\}$ indicates whether $\mathcal{V}$ accepts or rejects, and tr denotes a transcript exchanged between $\mathcal{P}$ and $\mathcal{V}$.*
- *$\mathcal{C}(x, \mathsf{tr})$: A compression algorithm that outputs a compressed ciphertext C.*
- *$\mathcal{R}(\mathsf{sk}, C)$: A receiver (or recovery) algorithm that outputs a plaintext w.*

VE is secure if it satisfies the following three properties.

Completeness VE_R is ϵ_{comp}-complete if for all $(x,w) \in R$.

$$\Pr\left[b \neq 1 \vee (x,w') \notin R \; : \; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{comp}}(\kappa)$$

Validity VE_R is ϵ_{val}-valid if for all pairs of PPT adversary $(\mathcal{A}^*, \mathcal{P}^*)$,

$$\Pr\left[b = 1 \wedge (x,w') \notin R \; : \; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}); \\ (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}^*(\mathsf{sk}), \mathcal{V} \rangle (\mathsf{pk}, x); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{val}}(\kappa)$$

Computational Honest Verifier Zero-knowledge VE_R is ϵ_{zk}-HVZK if there exits a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$, all $(x,w) \in R$,

$$\left| \Pr\left[i = i' \; : \; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ (b, \mathsf{tr}_0) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x); \\ \mathsf{tr}_1 \leftarrow \mathcal{S}(\mathsf{pk}, x); \\ i \xleftarrow{\$} \{0,1\}; i' \leftarrow \mathcal{D}(\mathsf{pk}, x, \mathsf{tr}_i); \end{array} \right] - \frac{1}{2} \right| \leq \epsilon_{\mathsf{zk}}(\kappa)$$

Note that computational HVZK (as opposed to perfect, or statistical) is the best possible in the context of verifiable encryption, as an unbounded adversary can always try $w' = \mathcal{R}(\mathsf{sk}, C)$ with all possible sk, checking whether $(x,w') \in R$.

Protocol 6: MPCitH-IOP$_R$

Parameters: The number of parties N; the number of parallel repetitions τ; the number of opened parties t; the challenge set $\mathsf{Ch} = \left\{\mathbf{e} \subset [N] \ : \ |\mathbf{e}| = t\right\}$.

Inputs: prover $\mathbf{P}$ receives (x, w); verifier $\mathbf{V}$ receives x.

Committing phase The first-round message of $\mathbf{V}$ is empty. $\mathbf{P}$ proceeds as follows.

1. Choose random $w_1, \ldots, w_N$ such that $w = \sum_{i=1}^{N} w_i$.
2. Emulate "in her head" the execution of Π_f on input $(x, w_1, \ldots, w_N)$.
3. Prepare, based on the execution, the share of the witness, and the randomness, the views $V_1, \ldots, V_N$ of the N parties; $\mathbf{P}$ outputs the proof string $\pi = (V_1, \ldots, V_N)$.

Query phase

1. $\mathbf{V}$ chooses a random $\mathbf{e} \in \mathsf{Ch}$ and queries the oracle for π with $\mathbf{e}$.
2. The oracle returns $(V_i)_{i \in \mathbf{e}}$.

Decision phase: $\mathbf{V}$ accepts if and only if $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$.

$\mathbf{P}$ and $\mathbf{V}$ execute τ instances of the above procedures in parallel. If $\mathbf{V}$ accepts in all τ executions, it outputs $b = 1$; otherwise it outputs $b = 0$.

5.2.3 MPC-in-the-Head Proofs as IOPs

In Protocol 6 we describe the blueprint of a generic MPC-in-the-head protocol characterized as a single-round IOP. The framework of IOPs allows for a modular design of ZK proof systems and is becoming increasingly common for constructing efficient SNARKs and MPC-in-the-head ZK proofs (e.g., [dOT21, CHM+20, CFF+20]). As in prior work, we first design an information-theoretically secure protocol in the form of an IOP, where commitments are idealized in that both hiding and binding hold unconditionally. This is why the security properties for IOPs are defined w.r.t. unbounded adversaries, and the computational assumptions will only come into play when we later compile the IOP into a verifiable encryption scheme via a cryptographic commitment scheme with extractability.

In MPCitH-IOP$_R$, $\mathbf{P}$ proves knowledge of a witness w such that $R(x, w) = 1$, where Π_f is an MPC protocol computing f that uses additive secret sharing over some finite field $\mathbb{F}$, and $R(x, w) := (f(w) \stackrel{?}{=} x)$. This protocol is similar to the one from [IKOS07] relying on the "idealized commitment functionality", but modified to cover MPC protocols with a broadcast functionality, so the prover may open $2 < t < N$ parties' views instead of two. We also employ the IOP framework following more recent MPC-in-the-head protocols such as Ligero [BFH+20] and Limbo [dOT21]. As we shall see below, as an IOP protocol it is straightforward to prove straight-line extractability of MPCitH-IOP$_R$. This will allow a smooth transition to SLE of the MPCitH proof systems we compile (with suitable commitment schemes), then to the validity of the resulting verifiable encryption schemes.

Our description also has parallel repetition: a simpler protocol is repeated τ times in parallel to increase soundness. These changes make presentation consistent with many practical MPCitH proof protocols (e.g., ZKB++, KKW and Banquet all use $(N-1)$-private MPC protocols with broadcast channels).

The helper function CheckView in MPCitH-IOP$_R$ takes the statement and a set of views as input and returns 1 if:

1. The outputs of the opened parties (determined by their views) are 1, and

2. The opened views are consistent with each other, with respect to x and Π_f,

and returns 0 otherwise. We further define a utility function GetW, which takes a party's view and extracts their share of the witness from it.

We also introduce the notion of k-consistency, which essentially guarantees N views form an honest run of Π_f, as long as for any k distinct subsets of party indices, the corresponding parties' views are consistent with each other. This generalizes the notion of pairwise consistency introduced previously in [IKOS07, Def. 2.2].

Definition 5.2 (k-consistency)**.** *A single repetition of the protocol* $\mathsf{MPCitH\text{-}IOP}_R$ *has* k-consistency *if for any x, for any set of views $(V_1, \ldots, V_N)$ and for any subset of the the challenge space $S \subseteq \mathsf{Ch}$ such that $|S| \geq k$, the following two conditions are equivalent:*

1. *for every $\mathbf{e} \in S$, $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$;*
2. *$(V_1, \ldots, V_N)$ form an honest execution of Π_f on a public input x and the corresponding per-party private inputs, $w_i = \mathsf{GetW}(V_i)$, are such that $x = f(\sum_{i \in [N]} w_i)$.*

Remark 5.1. *The above notion captures several different instantiations of MPC-in-the-head protocols. For example, the original protocol from [IKOS07, §3] opens 2-out-of-N parties (i.e., $t = 2$ in $\mathsf{MPCitH\text{-}IOP}_R$) and satisfies $\binom{N}{2}$-consistency because their Lemma 2.3 only guarantees the validity of N views as long as every possible pair of the views is consistent. ZKBoo and ZKB++ are essentially a special case of that protocol with $N = 3$ and therefore they have 3-consistency. Looking ahead, our DKGitH protocol in Section 5.5.1 works with N parties and the challenge set Ch is all subsets of $[N]$ of size $t = N - 1$.[2] We will show it satisfies 2-consistency thanks to the use of a broadcast functionality.*

Definition 5.3 (Canonical extractor)**.** *An extractor $\mathbf{E}$ for one repetition of $\mathsf{MPCitH\text{-}IOP}_R$ is called* canonical *if on input x and $\pi = (V_1, \ldots, V_N)$, it works as follows: $\mathbf{E}$ obtains witness shares via $w_i = \mathsf{GetW}(V_i)$ for $i \in [N]$ and then outputs a candidate witness $w := \sum_{i \in [N]} w_i$. For τ repetitions, the canonical extractor $\mathbf{E}^\tau$ runs $\mathbf{E}$ on each repetition $j \in [\tau]$ and outputs $w^{(j)}$ if $(x, w^{(j)}) \in R$ for some j, otherwise it outputs $\perp$.*

Now we prove knowledge error bounds for generic IOPs with k-consistency.

Lemma 5.2. *If a single repetition of $\mathsf{MPCitH\text{-}IOP}_R$ has k-consistency, then it is SLE with respect to the canonical extractor $\mathbf{E}$ with knowledge error $\epsilon_{\mathsf{sle\text{-}iop}} \leq \frac{k-1}{|\mathsf{Ch}|}$.*

Proof. Let V_i be the views output by a cheating prover $\mathbf{P}^*$ in the committing phase and $\mathbf{e} \in \mathsf{Ch}$ is the challenge sampled uniformly by the verifier $\mathbf{V}$ in the query phase. Further, let $w' = \sum_{i \in [N]} \mathsf{GetW}(V_i)$. Our goal is to bound the probability

$$\Pr\left[\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1 \wedge (x, w') \notin R\right] . \tag{5.1}$$

Define $\mathsf{GoodCh} := \left\{\mathbf{e} \in \mathsf{Ch} \ : \ \mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1\right\}$, i.e., a set of challenges that are accepting with respect to views $(V_i)_{i \in [N]}$ committed to by $\mathbf{P}^*$. If $|\mathsf{GoodCh}| \geq k$, then it must be that $(x, w') \in R$ due to k-consistency, so the canonical extractor always succeeds. If $|\mathsf{GoodCh}| < k$, then since $\mathbf{e}$ is sampled from Ch independently of V_i and thus of GoodCh as well, the probability that $\mathbf{e}$ falls in GoodCh is at most $\frac{k-1}{|\mathsf{Ch}|}$. □

[2] In practice, it suffices to send a single party index $\bar{i}$ whose view is *not* to be revealed.

One can easily generalize the above argument to deal with parallel repetitions. Proof is deferred to [TZ21].

Lemma 5.3. *If one repetition of* $\mathsf{MPCitH\text{-}IOP}_R$ *has* k*-consistency, then* τ *parallel repetitions are SLE with respect to the canonical knowledge extractor* $\mathbf{E}^\tau$ *with knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}} \leq \left(\frac{k-1}{|\mathsf{Ch}|}\right)^\tau$.

Remark 5.2. *We note that the above knowledge error is equivalent to the soundness error. For example, for ZKBoo and ZKB++ we have that* $k = 3$ *and* $\mathsf{Ch} = \{\{1,2\},\{2,3\},\{3,1\}\}$ *and therefore both the SLE knowledge error and soundness error are* $(2/3)^\tau$.

Finally, we recall the notion of t-privacy for an MPC protocol from [IKOS07]. We show t-privacy implies HVZK of the MPC-in-the-head IOP. Although we only consider the case of *perfect* t-privacy and HVZK, one can obtain a similar claim for statistical security of the lemma following the result of [IKOS07].

Definition 5.4 (t-privacy)**.** *The protocol* Π_f *is said to be* t*-private if there exists a PPT simulator* Sim *such that for every* $\mathbf{e} \in [N]$ *of size at most* t *and for every input* $(x, w_1, \ldots, w_N)$*, the joint view of parties in* $\mathbf{e}$ *is distributed identically to* $\mathsf{Sim}(\mathbf{e}, x, (w_i)_{i\in\mathbf{e}}, b)$ *where* $b = 1$ *if* $(x, \sum_{i\in[N]} w_i) \in R$ *and* $b = 0$ *otherwise.*

Lemma 5.4. *If the MPC protocol* Π_f *is* t*-private, then* $\mathsf{MPCitH\text{-}IOP}_R$ *is perfectly HVZK.*

Proof. An IOP simulator $\mathbf{S}$ takes x as input and proceeds as follows: (1) sample $\mathbf{e} \subset [N]$ of size t uniformly at random, (2) choose uniformly random witness shares w_i for $i \in \mathbf{e}$, (3) invoke $\mathsf{Sim}(\mathbf{e}, x, (w_i)_{i\in\mathbf{e}}, 1)$ to obtain joint views V_i for $i \in \mathbf{e}$, and (4) output $(\mathbf{e}, (V_i)_{i\in\mathbf{e}})$. This perfectly simulates the view of $\mathbf{V}(x)$ in the honest interaction with $\mathbf{P}$, since an honest $\mathbf{V}$ always queries the oracle with a set of party indices of size t and thus the t-privacy property guarantees perfect simulation of revealed views in the MPC execution. □

5.2.3.1 Protocols without k-consistency

While the notion of k-consistency has some generality and gives a convenient way to prove SLE of some three-round protocols, many MPC-in-the-head proof systems such as KKW and Banquet have challenge spaces not limited to party indices and therefore do not have k-consistency. However, we remark that they are easily checked to be straight-line extractable since $\mathbf{P}$ outputs per-party views that include the shares of the witness in the first round of the committing phase. The existing soundness analysis thus implies SLE of the corresponding IOP protocols. See [TZ21] for details.

5.3 Our Transform

In this section we present our transform, which generically constructs a verifiable encryption scheme MPCitH-VE from an MPCitH-IOP protocol in the class described in Protocol 6. We start with a simple construction of extractable commitments from public-key encryption, then come to our compiler in Section 5.3.2.

5.3.1 Extractable Commitments from Undeniable PKE

Given $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we consider the extractable commitment scheme $\mathsf{ECOM} := (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as defined in Section 5.2.1. As we shall see in later sections, IND-CPA security of PKE is not sufficient for guaranteeing validity of the resulting verifiable encryption, if the correctness is imperfect. The reason is that a malicious prover may be able craft a ciphertext c^* that can be correctly opened to plaintext m^* such that it passes validity checks performed by a verifier, while c^* decrypts to junk during the recovery phase. To prevent this attack, we require an additional property called *undeniability*. Intuitively, undeniability forces an adversary to open any ciphertext to the plaintext identical to the result of decryption. In Section 5.1.2 we discuss some similar (but different) notions from the literature.

Definition 5.5 (Undeniability)**.** *We say that a public-key encryption scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ϵ_{und}-undeniable *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa); \\ m \neq m' \ \wedge \ c = \mathsf{Enc}(\mathsf{pk}, m; r) \ : \ (c, m, r) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}); \\ m' := \mathsf{Dec}(\mathsf{sk}, c) \end{array}\right] \leq \epsilon_{\mathsf{und}}(\kappa)$$

The following utility lemma guarantees that an undeniable IND-CPA encryption scheme can be used as a secure extractable commitment with the simple construction given in 5.2.1.

Lemma 5.5. *If* PKE *is* ϵ_{und}*-undeniable and* ϵ_{cpa}*-IND-CPA secure, then the commitment scheme* ECOM *constructed from* PKE *is* ϵ_{cext}*-extractable with* $\epsilon_{\mathsf{cext}} \leq \epsilon_{\mathsf{und}}$*,* ϵ_{bind}*-binding with* $\epsilon_{\mathsf{bind}} \leq \epsilon_{\mathsf{und}}$ *and* ϵ_{hide}*-hiding with* $\epsilon_{\mathsf{hide}} \leq \epsilon_{\mathsf{cpa}}$*.*

Proof. We prove the three properties separately.

Extractability follows from undeniability. That is, if the adversary can output a tuple (c, m, r) breaking the extractability of ECOM, it also holds that $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and $m \neq \mathsf{Dec}(\mathsf{sk}, c)$. Therefore, (c, m, r) is also an instance breaking undeniability.

Binding follows from undeniability. Suppose there exists an adversary that outputs a tuple (m, r, m', r', c) such that it breaks binding with non-negligible probability, i.e., $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$.

Given such an efficient adversary $\mathcal{A}$ against the binding game, we construct another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break undeniability as follows.

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ as input, $\mathcal{B}$ forwards it to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs (c, m, r, m', r') such that $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$, the $\mathcal{B}$ first decrypts c: $\tilde{m} = \mathsf{Dec}(\mathsf{sk}, c)$ and proceeds as follows.
   a) If $\tilde{m} \neq m$, then $\mathcal{B}$ outputs (c, m, r) in the undeniability game.
   b) If $\tilde{m} \neq m'$, then $\mathcal{B}$ outputs (c, m', r') in the undeniability game.

Note that at least one of 2(a) or 2(b) must occur since $m \neq m'$. In either case, $\mathcal{B}$ successfully wins the undeniability game as long as $\mathcal{A}$ breaks binding. Clearly $\mathcal{B}$ succeeds with the same probability as $\mathcal{A}$, and $\mathcal{B}$'s runtime is the same as $\mathcal{A}$'s plus the cost of one Dec operation.

Hiding follows from the proof for Lemma 5.1, since it only relies on the IND-CPA security of PKE. □

5.3.1.1 How to construct undeniable PKE

Validity of our generic compiler described in the next section heavily relies on extractable commitments. The straightforward construction of ECOM requires undeniability, which is not necessarily satisfied by public-key encryption schemes with *statistical correctness*. As we shall see in [TZ21], this is not just a limitation in a security proof; a lack of undeniability actually allows cheating provers to break validity entirely. A natural question is whether one can generically add the undeniable property to *any* IND-CPA-secure encryption scheme with statistical correctness. We answer this question in the affirmative by proving that several variants of the Fujisaki–Okamoto transform [FO99, FO13, HHK17] can make a given PKE scheme undeniable in the random oracle model.

For example, suppose we are given an encryption function Enc that takes a public key, message, and random value as input, and a random oracle G that hashes into the randomness space of Enc. The simplest FO transform [FO99] defines Enc′ such that

$$\mathsf{Enc}'(\mathsf{pk}, m; r) := \mathsf{Enc}(\mathsf{pk}, m||r; \mathsf{G}(m||r)). \tag{5.2}$$

A crucial observation is that cheating provers are now forced to derive encryption randomness uniformly by querying the random oracle G. This makes it difficult to craft a malicious ciphertext c from biased randomness, which decrypts to a plaintext inconsistent with what she originally encrypted. Using the same observation we can also prove that well-known FO-based CCA conversion methods employed by Kyber and FrodoKEM achieve undeniability. Details are deferred to [TZ21].

5.3.2 Compiling MPCitH-IOP Into Verifiable Encryption

Our construction MPCitH-VE is given in Protocol 7. The description already incorporates the random subset optimization that will be analyzed in the next section. Here, we focus on the case of $n = \tau$ for simplicity. As for the intuition for our construction, we observed in Section 5.2.3 that for any MPCitH IOP following the [IKOS07] paradigm, there exists a (canonical) straight-line extractor that recovers the witness from the committed values of all parties. Recall that:

- The MPC protocol evaluates R with inputs x and w.
- The input x is public and w is shared amongst the parties.
- The view of each party must include their share of the witness and random tapes in order to allow verification to check consistency, since some of the outgoing messages of the parties must depend on both of these values.

Therefore, given the opening of the commitments of all parties (all N views), the extractor can recover the witness based on the shares of all parties. For constructing ZK proofs or signatures allowing for straight-line witness extraction, one can compile MPCitH-IOP by letting a prover commit to every per-party view with random oracle commitments as in [Pas03, KKW18, ZCD+20, DFMS21]: the extractor can reconstruct a witness by observing the RO query history. However, this does not suffice for instantiating verifiable encryption because the receiver (i.e., decryptor) in the real-world clearly has no access to the query history.

Our compiler takes an alternative approach similar to [Kat21, HLR21], which simultaneously realizes a straight-line extractable ZK proof system and valid verifiable encryption

Protocol 7: MPCitH-VE$_R$

Converts the MPCitH-IOP prover $\mathbf{P}$ and verifier $\mathbf{V}$ to an MPCitH-VE prover $\mathcal{P}$ and verifier $\mathcal{V}$ using the the extractable commitment scheme $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as constructed in Section 5.3.1.

Parameters: The number of parties N; the number of parallel repetitions τ; the number of opened parties t; the challenge set $\mathsf{Ch} = \left\{\mathbf{e} \in [N] \,:\, |\mathbf{e}| = t\right\}$; the subset size for compression n.

Key Generation $\mathcal{G}(1^\kappa)$: It invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^\kappa)$ and outputs $(\mathsf{pk}, \mathsf{sk})$.

Two-party protocol $\langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x)$**:**

1. $\mathcal{P}$ runs $\mathbf{P}$ on input (x, w) to obtain the proof string $\pi = (V_1, \ldots, V_N)$.
2. $\mathcal{P}$ separately commits to each of these N views to generate per-party commitments $(\mathsf{C}_1, \ldots, \mathsf{C}_N)$ where $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ and r_i is commitment randomness uniformly sampled from S_r.
3. $\mathcal{V}$ invokes $\mathbf{V}$ on input x to obtain challenge $\mathbf{e} \in \mathsf{Ch}$, and sends it to $\mathcal{P}$.
4. $\mathcal{P}$ opens the commitments of the t parties, by revealing $(V_i, r_i)_{i \in \mathbf{e}}$.
5. $\mathcal{V}$ sends the views $(V_i)_{i \in \mathbf{e}}$ to $\mathbf{V}$ as a response to the oracle query. It accepts if and only if:
   a) $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ and $r \in S_r$ for all $i \in \mathbf{e}$, i.e., $\mathcal{P}$ opened the views corresponding to $(\mathsf{C}_i)_{i \in \mathbf{e}}$ successfully, and
   b) $\mathbf{V}$ outputs 1.

$\mathcal{P}$ and $\mathcal{V}$ execute τ instances of the above protocol in parallel. If $\mathcal{V}$ accepts in all τ executions, it outputs $b = 1$ and a transcript

$$\mathsf{tr} = ((\mathsf{C}_i^{(j)})_{i \in [N]}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]} \,.$$

Otherwise, $\mathcal{V}$ outputs $b = 0$ and $\mathsf{tr} = \perp$.

Compression $\mathcal{C}(x, \mathsf{tr})$:

1. It samples a subset $S \subseteq [\tau]$ of size $n \leq \tau$ uniformly at random.
2. For $j \in S$, extract the t witness shares $w_i^{(j)} = \mathsf{GetW}(V_i^{(j)})$ for $i \in \mathbf{e}^{(j)}$ and partially reconstruct the witness $\tilde{w}^{(j)} = \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)}$.
3. Output the compressed ciphertext $C = (\tilde{w}^{(j)}, (\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}})_{j \in S}$.

Receiver $\mathcal{R}(\mathsf{sk}, C)$: To decrypt the ciphertext C, the receiver proceeds as follows.

1. For $j \in S$ and $i \notin \mathbf{e}^{(j)}$, extract the unopened parties' views $\hat{V}_i^{(j)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i^{(j)})$ and computes the corresponding witness shares $\hat{w}_i^{(j)} = \mathsf{GetW}(\hat{V}_i^{(j)})$. Let $w^{(j)} = \tilde{w}^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}$ be the jth candidate witness.
2. If there exits some $j \in S$ such that $(x, w^{(j)}) \in R$, output $w^{(j)}$. Otherwise, output $\perp$.

scheme. By replacing the commitment function with an *extractable* commitment ECOM (as defined in previous section) where the recipient has the decryption key sk, the recipient can decrypt the commitments to the unopened view(s) and recover all openings, then use the extractor algorithm to recover a witness. We remark that our transform naturally generalizes to other types of MPCitH protocols as well, since all such protocols (we are aware of) allow extraction of a witness given the openings of the per-party commitments (and indeed use this in their security reductions).

Because our presentation assumes the witness is shared with an additive secret sharing scheme, we make use of this to compress the ciphertext, by summing the t revealed shares into the single value $\tilde{w}$. If the secret sharing scheme of Π_f does not allow such partial reconstruction, then the ciphertext may simply include all shares. When generalizing to other types of secret sharing schemes the decryption operation must also be generalized to reconstruct w from the shares of all parties.

Theorem 5.1. *Let* $\mathsf{MPCitH\text{-}IOP}_R$ *be an MPC-in-the-head-based IOP in the class described by Protocol 6 that is perfectly HVZK and SLE with knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}}$*. Let* ECOM *be an extractable commitment scheme that has* ϵ_{cext}*-extractability and is* ϵ_{hide}*-hiding. Then the compiled protocol,* $\mathsf{MPCitH\text{-}VE}_R$ *described in Protocol 7 with* $n = \tau$*, is* ϵ_{val}*-valid with validity error* $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{sle\text{-}iop}} + \epsilon_{\mathsf{cext}}$*, and* ϵ_{zk}*-HVZK with* $\epsilon_{\mathsf{zk}} = \tau(N-t)\epsilon_{\mathsf{hide}}$*.*

HVZK directly follows from hiding of ECOM (and thus from IND-CPA of the underlying PKE). Proof of validity essentially proceeds as follows: if an MPCitH-VE cheating prover $\mathcal{P}^*$ can convince the verifier $\mathcal{V}$ while the receiver fails to decrypt a correct witness, then it must be that either (1) $\mathcal{P}^*$ broke extractability of ECOM, or (2) one can construct a pair of adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ that break SLE of $\mathsf{MPCitH\text{-}IOP}_R$. Adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ first extract views from the commitments sent by $\mathcal{P}^*$ and then forward them as a complete set of N views in the SLE-IOP game. Formal proof is deferred to [TZ21].

5.3.2.1 Optimizations

While the prover in our generic compiler MPCitH-VE commits to a complete per-party view V_i using ECOM, several standard optimization techniques in the literature also are applicable in our setting for better computational and communication complexities. Notice that $\mathcal{R}$ would only need witness shares $(w_i)_{i\in[N]}$ to be able to recover the plaintext. Hence, it would be sufficient to have the prover $\mathcal{P}$ commit to w_i using ECOM, and to the rest of the strings in V_i using the random oracle commitments as the ZKBoo/ZKB++ prover does [GMO16, CDG+17]. Since ECOM is instantiated with PKE in practice while the RO is instantiated with cryptographic hash functions, this would significantly reduce the size of transcripts and could save both prover and verifier time for creating/opening commitments.

Moreover, following [KKW18], in case the MPC protocol Π_f relies on a broadcast channel and thus $N-1$ out of N views are revealed, we can decouple broadcast messages $(\mathsf{msgs}_i)_{i\in[N]}$ from per-party views to reduce the communication complexity, where each msgs_i consists of messages broadcast by party i. That is, the prover $\mathcal{P}$ first generates a root seed sd^* to derive per-party seeds $(\mathsf{sd}_i)_{i\in[N]}$ with a binary tree construction. $\mathcal{P}$ now only commits to each seed sd_i used for deriving a witness share and a random tape of party i using ECOM, and sends $h = \mathsf{H}((\mathsf{msgs}_i)_{i\in[N]})$. On receiving challenge $\bar{i} \in [N]$ from $\mathcal{V}$, indicating the index of unopened party, $\mathcal{P}$ reveals $\mathsf{msgs}_{\bar{i}}$ and $\lceil \log_2(N) \rceil$ nodes in the tree, which are sufficient to compute $(\mathsf{sd}_i)_{i\in[N]\setminus\{\bar{i}\}}$. From such information, $\mathcal{V}$ can reconstruct the

remaining broadcast messages, check h against broadcast messages sent by all N parties, and check that $N-1$ parties on input $(\mathsf{sd}_i)_{i\in[N]\setminus\{\bar{i}\}}$ lead to a correct output with respect to x and $\mathsf{msgs}_{\bar{i}}$.

Our DKG-in-the-head protocol in Section 5.5.1 benefits from these optimizations.

5.3.3 Compiling Banquet and KKW

Although the IOPs corresponding to KKW and Banquet (given in [TZ21]) are not exactly in the class described by MPCitH-IOP, we can compile them into verifiable encryption schemes using essentially the same idea.

To compile Banquet-IOP, it is sufficient to have the VE prover $\mathcal{P}$ commit to the per-party seeds $(\mathsf{sd}_i)_{i\in[N]}$ with an extractable commitment scheme during the first round. The second and third round operations are identical to the original Banquet-IOP protocol, and the VE verifier $\mathcal{V}$ proceeds by following the decision phase of Banquet-IOP and accepts iff $\mathbf{V}$ accepts and the $N-1$ per-party commitments are opened correctly. The compression and receiver algorithms $\mathcal{C}$ and $\mathcal{R}$ are defined analogously to those of MPCitH-VE, except that the witness offset Δw is added by $\mathcal{C}$ when creating a partially reconstructed witness $\tilde{w}$. Since the receiver tries to decrypt by using the SLE extractor algorithm defined in [TZ21], the compiled protocol has ϵ_{val}-validity with $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{cext}} + \epsilon_{\mathsf{sle}}$, assuming ϵ_{cext}-extractability of ECOM and ϵ_{sle}-SLE of Banquet-IOP.

Likewise, we can compile KKW-IOP by having the VE prover $\mathcal{P}$ commit to the offline per-party states $(\mathsf{st}_i^{(j)})_{i\in[N]}$ with ECOM. On the other hand, the other commitments in KKW-IOP can be instantiated with the usual random oracle commitments as in the original KKW protocol. As we only need τ revealed online executions to recover a witness, the compression algorithm $\mathcal{C}$ outputs as a ciphertext $\tilde{w}^{(j)} = \sum_{i\neq\bar{i}_j} \lambda_i^w \oplus \hat{w}^{(j)}$ and $\mathsf{C}_{\bar{i}_j}^{(j)}$ for $j \in T \subset [M]$, where each witness mask share λ_i^w is obtained from the revealed value $\mathsf{st}_i^{(j)}$. Then the receiver $\mathcal{R}$ extracts the unopened share of the witness mask from $\mathsf{C}_{\bar{i}_j}^{(j)}$ and XORs it with $\hat{w}^{(j)}$ to recover a candidate witness.

5.3.4 Applying Fiat–Shamir

Following the standard Fiat–Shamir transform [FS87], we can make our verifiable encryption protocol MPCitH-VE non-interactive in the random oracle model, by hashing the first prover messages together with x and pk to obtain the challenge $\mathbf{e} \in \mathsf{Ch}$. Since the base interactive protocol has three rounds, the FS transform introduces a multiplicative factor of q security loss in validity, where q is the number of random oracle queries made by a non-interactive cheating prover. Note that this loss is well-known in (knowledge) soundness analysis for FS-NIZK proofs and EUF-KOA security of signatures constructed from canonical identification schemes [KMP16]. Formal analysis is deferred to [TZ21]. Banquet-based verifiable encryption however requires a separate concrete analysis dedicated to the non-interactive version, since it has 7 rounds of interaction. Because the EUF-KOA security analysis of Banquet as a signature scheme [BdK+21a, Theorem 2] already evaluates the probability that the witness (i.e., secret signing key) extraction fails, their analysis can be reused in large part to derive the concrete validity error of non-interactive Banquet-VE. Construction of Banquet-NIVE and validity analysis are deferred to [TZ21].

5.3.5 Achieving Strong Validity

To the best of our knowledge, prior definitions of validity for verifiable encryption in the literature assume that the key generation phase is always performed honestly. One can strengthen the validity property so that a cheating prover takes control of key generation. Formally, we say a VE scheme has ϵ_{sval}-*strong validity* if for all pairs of PPT adversary $(\mathcal{A}^*, \mathcal{P}^*)$,

$$\Pr\left[\begin{array}{rr} b = 1 \wedge & (x, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{A}^*(1^\kappa); \\ (x, w') \notin R \wedge : & (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}^*(\mathsf{sk}), \mathcal{V} \rangle (\mathsf{pk}, x); \\ (\mathsf{pk}, \mathsf{sk}) \in \mathcal{G}(1^\kappa) & C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array}\right] \leq \epsilon_{\mathsf{sval}}(\kappa).$$

We remark that allowing $\mathcal{A}^*$ to choose $(\mathsf{pk}, \mathsf{sk})$ is very strong, and that in practice it's not possible to check whether $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{G}(1^\kappa)$. However, without this condition, note that $\mathcal{A}^*$ can trivially break strong validity by generating a keypair then setting sk to 0. In the context of our verifiable key backup scenario, the device could be encrypting the key to a future instance of itself, or to another device in the same security domain. Here the user must trust that the device importing the key has generated its keypair honestly. This seems to be the best possible validity assurance when the device is responsible to store sk.

If ECOM is instantiated with a perfectly correct PKE, we can achieve strong validity of MPCitH-VE. Observe that if PKE has perfect correctness, then for every key pair and for every ciphertext, the corresponding plaintext is uniquely determined. Therefore, as long as the key pair is in the right domain (which the verifier can easily check) undeniability can never be broken regardless of the distribution of keys.

5.4 Methods for Compressing Ciphertexts

Because MPCitH protocols use τ parallel repetitions to boost soundness, the ciphertexts output by our transform can be large. For example, for 128-bit security, τ could range from 20 to 219. Each repetition outputs one PKE ciphertext and a share of the witness, so the total size is $\tau(|\mathsf{PKE.Enc}| + |w|)$. Also, in the post-quantum PKE case, lattice-based constructions can have relatively large ciphertexts. An interesting question is whether these can be compressed, since these ciphertexts will usually be very redundant: note that for an honestly created proof all τ repetitions encrypt the same witness (in different ways), and the receiver will only need to decrypt one.

In this section we give two methods to compress the verifiable encryption ciphertexts output by schemes created with our transform. The first, called the *random subset method*, is very simple, incurs no computational overhead, and can reduce ciphertext size by a factor of three when τ is large. The second approach, called the *equality proof method*, is optimal as it achieves constant size ciphertexts, $O(|w|)$ (provided PKE has constant ciphertext expansion). However, it requires special properties of PKE, increases proof size, prover and verifier computational costs significantly, so it is more of a possibility result rather than a practical construction. We highlight improving compression as an interesting direction for future work.

5.4.1 The Random Subset Method

This compression method is rather simple, but the impact on ciphertext size can be significant, and the cost to the prover is nothing, and almost nothing to the verifier. That

is, we set $n < \tau$ in Protocol 7 to optimize the compression and receiver algorithms. Upon receiving a verifiable encryption proof with our transform, the verifier has a set of τ ciphertext components, corresponding to the τ parallel repetitions used to produce the proof. The verifier chooses a subset of the ciphertexts to keep at random, and discards the others. The size of the subset is denoted n, and is a parameter of the method.

We stress that soundness of the proof is unchanged, since the entire is proof is communicated to the verifier and checked. Only the analysis of the validity error must be updated, since the receiver now has only n ciphertexts.

Let s be the number of ciphertexts in the initial set of size τ that are bad, meaning they do not decrypt to the witness. For the proof systems we consider, having $s > 0$ is quite easy, as it only requires guessing a small part of the challenge. Note that s must be at least n, otherwise the attack against compression never succeeds, since V's output always contains one or more valid ciphertexts.

Below we will choose parameters for the random subset method applied to different proof systems, in the interactive case. The adversary $\mathcal{P}^*$, is a cheating prover who knows the witness, and tries to create a verifiable ciphertext where decryption fails. Then the general form of $\mathcal{P}^*$'s success probability is

$$\begin{aligned}
&\Pr\left[\mathcal{C} \text{ selects } n \text{ of } s \text{ bad ctexts } \wedge\ \mathcal{V} \text{ accepts a proof with } s \text{ bad ctexts}\right] \\
&= \frac{\#\text{subsets with } n \text{ bad ctexts}}{\#\text{ of subsets}} \cdot \Pr\left[\mathcal{V} \text{ accepts a proof with } s \text{ bad ctexts}\right] \\
&= \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot (\epsilon_{\mathsf{sle\text{-}iop}}(s) + \epsilon_{\mathsf{cext}})
\end{aligned}$$

where $\epsilon_{\mathsf{sle\text{-}iop}}(s)$ is the probability that an IOP prover wins the SLE-IOP game with s parallel repetitions, and "ctexts" is short for ciphertexts. A more formal analysis is given in [TZ21], where we prove the following theorem.

Theorem 5.2. *Let* $\mathsf{MPCitH\text{-}IOP}_R$ *be an MPC-in-the-head-based IOP in the class described by Protocol 6 with SLE knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}}$*. Let* ECOM *be an extractable commitment scheme with* ϵ_{cext}*-extractability. Then* $\mathsf{MPCitH\text{-}VE}_R$ *is* ϵ_{val}*-valid with validity error*

$$\epsilon_{\mathsf{val}} = \max_{n \le s \le \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot (\epsilon_{\mathsf{sle\text{-}iop}}(s) + \epsilon_{\mathsf{cext}})\,.$$

Generally, the amount of compression possible is larger when τ is larger, as demonstrated by the ZKB++ example (where $\tau = 219$ for 128-bit security). The DKGitH example requires much smaller τ (in the range 16–32), and compression is limited, or none at all. However, we can increase τ to larger values than strictly necessary, in order to compress the ciphertext further, see Fig. 5.2 for a range of options with fixed N and the first row of Table 5.1 for a concrete example. This reduces ciphertext size at the expense of proof size, which can be beneficial in applications that check the proof then discard it, but store the ciphertext.

5.4.1.1 Application to IKOS/ZKBoo/ZKB++

We consider interactive IKOS-style protocols, such as ZKBoo and ZKB++. For each repetition of the protocol, they have $\binom{N}{2}$-consistency, where N is the number of parties. As ZKBoo and ZKB++ have $N = 3$ and $\mathsf{Ch} = \{1, 2, 3\}$ they have 3-consistency and thus

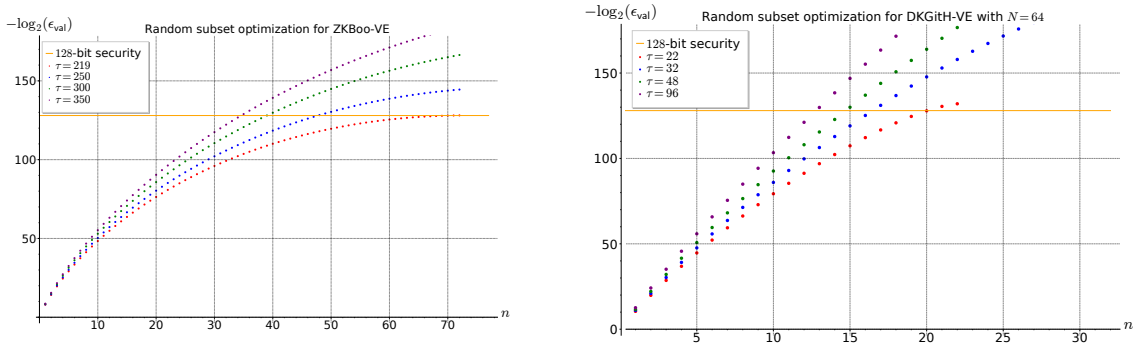

Figure 5.2: Approximate minimum cost of breaking validity of ZKBoo-based VE (left) and DKGitH-based VE (right) with a random subset of size n. The parameter τ denotes the number of parallel repetitions. The number of parties N is fixed to 3 for ZKBoo and 64 for DKGitH, respectively. Note that $\tau = 219$ corresponds to the `picnic-L1` parameters from the Picnic spec [ZCD+20].

are SLE with knowledge error $\epsilon_{\text{sle-iop}}(s) \leq 2/3$ from Lemma 5.2. Hence, if the verifier outputs all τ ciphertexts, the validity error is $\epsilon_{\text{val}}(\tau) \leq (2/3)^\tau + \epsilon_{\text{cext}}$. If the random subset optimization is applied, however, this will give cheating provers an extra strategy to break validity: by breaking soundness only in s executions and performing the remaining $\tau - s$ runs honestly using the genuine witness, the receiver in the validity game still fails to obtain the right witness if the subset of size n is selected entirely from the s "bad" instances. Hence, now the validity error can be calculated as

$$\epsilon_{\text{val}}(\tau, n) = \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\left(\frac{2}{3} \right)^s + \epsilon_{\text{cext}} \right).$$

In Fig. 5.2 we show the costs of breaking validity for different combinations of τ and n assuming ϵ_{cext} is negligible. We see that $n = 70$ provides 128-bit security with $\tau = 219$ repetitions, meaning we can compress ciphertexts by a factor 3 at no cost. If we increase τ slightly to 250 (meaning proof size and prover/verifier time increase by roughly 1.14x) then we can set $n = 50$ and compress ciphertexts by a factor 4.4.

5.4.1.2 Application to DKGitH

This is similar to IKOS, except that the default soundness error is different. Because the corresponding MPC protocol uses a broadcast functionality, the prover reveals $N - 1$ parties' views and thereby the knowledge error is at most $1/N$, instead of $1 - 1/\binom{N}{2}$. Hence, for τ parallel repetitions we have

$$\epsilon_{\text{val}}(\tau, n, N) = \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\left(\frac{1}{N} \right)^s + \epsilon_{\text{cext}} \right). \tag{5.3}$$

In Fig. 5.2 we show the costs of breaking validity for different combinations of τ and n assuming ϵ_{cext} is negligible. As τ is smaller, the amount of compression we get for free is limited to only 2 ciphertexts (i.e., we can set $n = 20$ when $\tau = 22$). The option of increasing τ is again possible, but provides less compression and at a higher cost. In addition to the choices of (n, τ) given in Fig. 5.2, Table 5.1 gives some concrete examples showing proof and ciphertext size along with estimates of the prover and verifier times.

5.4.2 The Equality Proof Method

Recall that in a VE scheme created with our compiler, decryption iterates over the component ciphertexts (from each parallel repetition) until the reconstruction function recovers a witness. It is guaranteed that at least one of the component ciphertexts will cause decryption to succeed.

In an honestly generated proof, all component ciphertexts are valid, and decryption will always succeed on the first attempt. If after the VE protocol, the prover were able to additionally prove that $\mathcal{R}$ would output the same witness from all of the component ciphertexts, then the verifier could keep only one of the component ciphertexts, making the VE ciphertext constant size. This is because either: all values are equal and correct, or all values are equal and incorrect, but the latter case is equivalent to creating an invalid proof, which is possible with only negligible probability by soundness of the proof protocol.

Note that the equality proof proves that $\mathcal{R}$ outputs the same value for all component ciphertexts – *and is not requiring that we prove the relation.* The crux of $\mathcal{R}$ for MPCitH protocols is recombining additive shares of the witness, a comparatively simple operation. However one of the shares is encrypted, meaning we are back to proving something about encrypted data. We describe one instantiation of the idea to show that this is possible without resorting to general methods, by using PKE in a non-black-box way.

Theorem 5.3. *Let Π be an MPCitH-based IOP in the class given by Protocol 6 with $t = N - 1$, for a relation R where $|w| = \kappa$. Then there exists a VE scheme Π' with a compression algorithm that produces $O(\lambda)$ ciphertexts for Π', assuming Paillier's encryption scheme is IND-CPA secure.*

Sketch. We describe the construction of the verifiable encryption scheme Π'. First we compile Π to a VE scheme using a slight variant of Protocol 7. Namely, we split the per-party commitments into two so that the share of the witness and other information in the view are committed separately. Thus we have an additional commitment public key pk' for a second extractable commitment scheme Commit' (which may be the same as Commit, or a more efficient hash-based scheme, extractable in the ROM, since extraction won't be required for decryption). Then $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ is instead computed as $\mathsf{C}_i = (\mathsf{Commit}(\mathsf{pk}, w_i; r_i), \mathsf{Commit}'(\mathsf{pk}', v_i; s_i))$, where (wlog) each view is assumed to be $V_i = w_i || v_i$. Additionally, we assume that w is shared with XOR, so the shares are κ-bit strings.

Next, Π' is instantiated with extractable commitments constructed from the Paillier encryption scheme. Paillier is IND-CPA secure under the decisional composite residuosity assumption [Pai99], and encryption is perfectly correct, so it is a secure commitment scheme by Lemma 5.1. Further, each bit of the witness share is encrypted separately which will allow bitwise operations using the homomorphic properties of Paillier encryption.

The new compression algorithm $\mathcal{C}'$ requires input from $\mathcal{P}$ (the equality proof) and $\mathcal{V}$ runs it to check the proof and keep the final ciphertext. The steps for $\mathcal{P}$ are:

1. Run the compression algorithm $\mathcal{C}$ from Protocol 7, to get the set $C = (\tilde{w}^{(j)}, \widehat{\mathsf{C}}^{(j)})_{j\in[\tau]}$, where $\widehat{\mathsf{C}}^{(j)}$ is the unopened commitment for the jth parallel execution. Since $t = N-1$ there is only one commitment per parallel repetition.
2. Recall that $\widehat{\mathsf{C}}^{(j)} = \mathsf{Enc}(\mathsf{pk}, \hat{w}^{(j)})$ and $w = \hat{w}^{(j)} \oplus \tilde{w}^{(j)}$. Using the additive homomomorphic property of encryption, compute $C' = (\mathsf{Enc}(\hat{w}^{(1)} \oplus \tilde{w}^{(1)}), \ldots, \mathsf{Enc}(\hat{w}^{(\tau)} \oplus \tilde{w}^{(\tau)}))$ as described in [TZ21]. This is possible because $\tilde{w}^{(j)}$ are public constants, and there

is only one unopened party, so we only need to compute the XOR of one public and one encrypted value.

3. Convert the set C' of bitwise encryptions of w, to the set C'' of integer encryptions of w as described in [TZ21]. This is again possible using the homomorphic property, by computing $w = \sum_{i=0}^{\kappa} 2^i w_i$ (converting from binary to integer), and choosing paramters such that κ-bit strings fit in the plaintext space of Enc.
4. Prove all ciphertexts in C'' have the same plaintext. This step can be realized, e.g., with a standard generalization of Schnorr's proof of knowledge of a discrete logarithm (details in [TZ21]). A non-interactive equality proof π is output by $\mathcal{P}$ and sent to $\mathcal{V}$.

The verifier $\mathcal{V}$ repeats Steps 1-3 to compute C'', then checks that π is valid. If so, $\mathcal{V}$ outputs the first ciphertext in C'' as the encryption of w.

Since the output compression is one ciphertext, the resulting VE ciphertext clearly has size $O(\lambda)$.

In terms of security, the protocol until Step 2 of $\mathcal{C}'$ is secure by Theorem 5.1, since the modifications to the commitment scheme maintain the required extractability and hiding properties. For the next part of $\mathcal{C}'$, we argue that the plaintext transforms in Steps 2 and 3 to compute C'' are 1:1 and thus maintain validity. Because Theorem 5.1 guarantees that the scheme is valid, decryption of C succeeds with overwhelming probability, which means that at least one component ciphertext is an encryption of w that is valid, in particular the plaintexts are guaranteed to be single bits. When a ciphertext in C is an encryption of individual bits, then steps 2 and 3 are reversible, implying that if ciphertext j in C is valid, then ciphertext j in C'' is also valid. Finally, as argued above since C'' contains at least one valid encryption of w, all ciphertexts must encrypt w assuming the equality proof in Step 4 is sound with overwhelming probability. The assumptions required for the proof in Step 4 can vary, but with an interactive version of Schnorr's proof we need only assume that discrete logarithms are hard in the Paillier group, which is implied by the DCR assumption required for security of Paillier encryption. □

The construction has drawbacks that keep it from being practical, and it would be interesting to address them. Because we require some (relatively weak) homomorphic properties, we lose the flexibility in the choice of PKE, and a suitable PQ-secure instantiation requires investigation. The cost of creating and communicating of the proof soars because we require bitwise encryptions of witness shares, meaning the prover must compute $O(\tau N \lambda)$ individual Paillier encryptions. In practice this cost could be significantly reduced by using bitwise exponential Elgamal, however then the final ciphertext would have to remain in the bitwise representation (to allow efficient decryption) meaning the ciphertext would have size $O(\lambda^2)$, rather than $O(\lambda)$.

5.5 Concrete Instantiations

In this section we give some example instantiations for discrete logarithms, RSA and AES to estimate the concrete performance of verifiable encryption realized with our transform.

Interactivity All of the performance estimates are given for the non-interactive versions of proofs. However, we note that it is also possible in many applications (such as in verifiable key backup) where the verifier will only accept a small number of failed attempts by a prover, to use an interactive proof with 40–64 bits of interactive security (analogous to the

case of interactive identification schemes [FS87, Section 2.3]). For all the MPCitH protocols we consider this reduces number of parallel repetitions significantly, in turn reducing the prover and verifier time, proof size and ciphertext size by a factor 2–3.

5.5.1 Verifiable Encryption of Discrete Logs in Prime Order Groups

Perhaps the most fundamental relation in cryptography is the discrete logarithm in a prime order group $\mathbb{G}$, i.e., (y, x) such that $y = g^x$ where $\langle g \rangle = \mathbb{G}$. As an application our transform, we give a new protocol to verifiably encrypt a discrete logarithm. We construct an MPC protocol to compute y from shares of x, which naturally gives an MPCitH protocol to prove knowledge of x. When compared to the most efficient proof of knowledge for discrete logarithms, the Schnorr proof, our new protocol is much less efficient, but it is amenable to our transform, and can therefore be used to verifiably encrypt discrete logs. We can then verifiably encrypt DH, ECDH, DSA and ECDSA keys directly as key pairs for these algorithms are discrete logs, and in Section 5.5.1.2 we explain how this scheme can also be used to encrypt RSA keys.

As an aside, we remark that our new proof protocol has a tight reduction to the discrete logarithm problem in the random oracle model. This feature is of theoretical interest as it implies a signature scheme based on the discrete logarithm problem with a tight security reduction.

Baselines for Comparison We compare to two protocols from the literature. The first is the Camenisch-Damgård protocol [CD00] for a generic Σ protocol, combined with Schnorr's Σ-protocol [Sch91] for discrete logs with binary challenges. This is the only verifiable encryption scheme we are aware of that works for discrete logarithms in any cyclic group, and allows a flexible choice of PKE (as our protocol does). It also requires the random oracle assumption to make the proof non-interactive.

The second, more efficient, protocol in [CD00] paper has k parallel repetitions, and the verifier selects a subset to form the output, and audits the encryption step of the $k - u$ other repetitions (and the verifier checks all repetitions have a valid transcript for the Σ protocol with one challenge). No parameters are given for concrete, non-interactive security – we found that for κ-bit security, (k, u) must be chosen so that $\binom{k}{u} \geq 2^\kappa$. Then there are multiple possible choices for (k, u), which trade ciphertext size for computation: we can have a small decrease in ciphertext size, for a large increase in computation and proof size. Our comparison in Table 5.1 gives some of the options.

Another VE scheme we compare to is from [NRSW20], which can encrypt a discrete logarithm in an elliptic curve group, using a special PRF called Purify. The scheme does allow, e.g., encryption of an ECDSA private key, but requires that encryption be done with an Elgamal-like PKE. A complication related to implementation of the Purify PRF is that one must choose an additional pair of elliptic curves, related to the group order of the curve where the discrete logarithm is defined, such that the DDH assumption holds. In addition to making these additional parameter choices, we must also make an assumption beyond the DLP + PKE assumptions in $\mathbb{G}$ (as in [CD00] and our scheme).

We omit comparison to [CS03] since it only works for discrete logarithms in a group suitable for Paillier's encryption scheme, and the PKE is fixed to Paillier's scheme as well. The scheme is not suitable for encrypting an ECDSA private key, one of our motivating examples.

5.5.1.1 The proof protocol: DKG-in-the-head

We first describe the base non-interactive ZK proof system DKGitH for relation $R = \left\{(y, x) \ : \ y = g^x\right\}$. The core idea of the protocol is based on the additive homomorphism of private keys, under multiplication of public keys, and may be folklore (an early reference describing it is [Ped92]). To compute $f(x) = g^x \stackrel{?}{=} y$ in a distributed manner, the prover $\mathcal{P}$ provides shares of x to the N parties such that $x = \sum_{i=1}^{N} x_i \pmod p$. Then $\mathcal{P}$ emulates a simple distributed key generation (DKG) protocol Π_f that proceeds as follows.

1. Each party i computes $y_i = g^{x_i}$, and broadcasts y_i.
2. Output $y = \prod_{i=1}^{N} y_i$

$\mathcal{P}$ commits to the shares of the parties, and the y_i values (together these two values makeup party P_i's view), then the verifier $\mathcal{P}$ selects one party to remain unopened, having index $\bar{i}$. In the response, the prover sends the views of the other $N-1$ parties, along with $y_{\bar{i}}$, and a commitment to $x_{\bar{i}}$. Based on the revealed values, $\mathcal{V}$ checks that $y = y_{\bar{i}} \prod_{i \in [N], i \neq \bar{i}} g^{x_i}$ and that each y_i is computed correctly.

The protocol Π_f is perfectly $(N-1)$-private: suppose we are given the index of a party $\bar{i}$, we show that we can simulate the views of the other $N-1$ parties, such that simulated and real transcripts are perfectly indistinguishable. First the simulator chooses x_i at random, for $i \neq \bar{i}$ and computes $y_i = g^{x_i}$, as in the real protocol. Then for party $\bar{i}$, the simulator sets $y_{\bar{i}} = y/(\prod_{i \neq \bar{i}} y_i)$. Note that

$$y_{\bar{i}} = g^{x - \sum_{i \neq \bar{i}} x_i}, \text{ and } x_{\bar{i}} = x - \sum_{i \neq \bar{i}} x_i$$

are distributed exactly as in the real protocol.

Along with the core idea, the full protocol DKGitH specified in [TZ21] uses two ideas (originating in [KKW18]) that are now standard in protocols of this type. First, the shares of the parties are computed by reading random values from their tapes, and the first share is corrected with an auxiliary value that depends on the secret. Second, the tapes are derived from a per-iteration seed with a binary tree construction, so that the $N-1$ revealed seeds can be opened more efficiently by revealing $\lceil \log_2(N) \rceil$ seeds.

Remark 5.3. *Although* DKGitH *already incorporates the optimizations described in Section 5.3.2.1, the underlying* MPCitH-IOP *protocol instantiated with* Π_f *does satisfy the requirements from Section 5.2.3 so that our general compiler theorem applies: the challenge space is* $\mathsf{Ch} = \left\{\mathbf{e} \subset [N] \ : \ |\mathbf{e}| = N-1\right\}$*; party* i*'s view* V_i *consists of* $(x_i, (y_{i'})_{i' \neq i})$*; the function* $\mathsf{GetW}(V_i)$ *outputs* x_i*; the function* $\mathsf{CheckView}(y, (V_i)_{i \in \mathbf{e}})$ *parses* V_i *as* $(x_i, (y_{i',i})_{i' \neq i})$ *and checks* $y \stackrel{?}{=} g^{x_i} \prod_{i' \neq i} y_{i',i}$ *for all* $i \in \mathbf{e}$ *and* $y_{i',i} \stackrel{?}{=} g^{x_{i'}}$ *for* $i \in \mathbf{e}$ *and* $i' \in [N] \setminus \{i, \bar{i}\}$*, where* $\bar{i} \notin \mathbf{e}$ *is the index of the unopened party. We can prove the protocol has (i) 2-consistency (Definition 5.2) and thus is (ii) SLE with* $\epsilon_{\mathsf{sle\text{-}iop}} = 1/N$ *(Lemma 5.2). Showing the condition 1. from 2. in Definition 5.2 is trivial. To show the converse, let* $\mathbf{e}, \mathbf{e}'$ *be two distinct challenges. If* CheckView *outputs 1 w.r.t. both challenges, then for some* i *such that* $i \in \mathbf{e} \cap \mathbf{e}'$*, it must be that* $y = g^{x_i} \prod_{i' \neq i} g^{x_{i'}} = g^{x_1 + \ldots + x_N}$*. Hence,* $(V_1, \ldots, V_N)$ *form an honest execution of* Π_f *on* y *and witness shares* $(x_i)_{i \in [N]}$ *as inputs.*

Applying our transform In [TZ21] we describe how chose parameters for 128-bit concrete security, describe the hashed Elgamal PKE we use, and describe the optimizations we apply

Scheme	N	τ	n	k	u	$\lvert\mathsf{tr}\rvert$	$\lvert C\rvert$	(RS)	$\mathcal{P}$ exp.	(ms)	$\mathcal{V}$ exp.	(ms)
DKGitH	64	48	15			7 744	1 536	(480)	6 144	(239.62)	6 048	(235.87)
	85	20	20			3 584	640	(640)	3 400	(132.60)	3 360	(131.04)
	16	32	30			4 160	1 024	(960)	1 024	(39.94)	960	(37.44)
	4	64	48			6 208	2 048	(1 536)	512	(19.97)	384	(14.98)
[CD00]				712	20	35 100	1 922		2 880	(112.32)	2 880	(112.32)
				250	30	13 500	2 884		1000	(39.00)	1000	(39.00)
				132	64	9 424	6 152		528	(20.59)	528	(20.59)
[NRSW20]						1100	64		24 823	(968.10)	1 316	(51.32)

Table 5.1: Parameters and performance estimates for verifiable encryption of a discrete logarithm. Our scheme is in the first part of the table, followed by the generic scheme from [CD00] (combined with Schnorr's proof protocol [Sch91]), followed by the Purify PRF-based construction of [NRSW20]. Sizes are given in bytes. The ciphertext size for our scheme when the random subset (Section 5.4.1) compression method is used is given in parenthesis.

once these choices are fixed. We then explain how we obtained the size and speed estimates used in this section.

Examples and comparison We give some concrete parameters showing various time-speed trade offs, and compare to related work in Table 5.1. We give three parameter sets, and estimate the size in bytes of the transcript tr, the VE ciphertext $|C|$ (both with and without random subset (RS) compression), as well as the computational costs of the prover and verifier. For the costs we count the number of exponentiations, and for reference also give an estimated time in milliseconds (ms) by using the timings given in [NRSW20] (based on their benchmarks from an Intel i7-7820HQ system pinned to 2.90 GHz).

The options for our scheme offer short ciphertexts (480–640 bytes), at the expense of higher prover and verifier times, or much lower times, but with larger ciphertexts (1536 bytes) and proof sizes, or somewhere in the middle.

When compared to [NRSW20], there the ciphertext size is a regular Elgamal ciphertext, the proof size is about 1KB, but the prover and verifier times are 943ms and 50ms respectively. Notably, the prover time is much lower with our scheme, about 10-24x faster with the parameters shown.

We also compare to the [CD00] scheme with Schnorr's Σ-protocol for discrete logs, using the same hashed Elgamal scheme. It has proof size $\kappa + k\ell_p + (k-u)\kappa + u\ell_C$ and ciphertext size $u(\ell_C + \ell_p + 1)$. Prover and verifier must compute $4k$ exponentiations. In terms of proof and VE ciphertext size, our scheme always outperforms [CD00]. The running time of the prover and verifier are nearly the same.

5.5.1.2 Verifiable Encryption with RSA

There are two obvious ways to generalize the DKG-in-the-head idea to the RSA setting. First, we can verifiably encrypt an RSA private exponent d, by using the above proof of a discrete logarithm to prove knowledge of d such that $(m^e)^d = m \pmod n$, where (e, n) is an RSA public key and m is an arbitrary value. Thus we can efficiently verifiably encrypt RSA encryption and signing keys.

Second, we can prove knowledge of a preimage of a one-way group homomorphism. For example, if the homomorphism is $\phi : m \mapsto m^e \bmod n$ with $n = p \cdot q$, one can design a simple MPCitH protocol for knowledge of an RSA preimage: the parties share m multiplicatively, $m = m_1 \cdots m_N \pmod n$ then broadcast $\phi(m_i) = m_i^e$, and then check that $c = \prod m_i^e \pmod n$. This can be used to prove knowledge of an RSA plaintext corresponding to a given ciphertext (a more direct type of verifiable encryption), or knowledge of a message corresponding to a given signature. The MPC protocol can easily be extended to prove additional properties of m as well.

5.5.2 Verifiable Encryption of AES Keys

With our transform applied to Banquet-IOP, one can verifiably encrypt an AES private key used for generating a given public ciphertext. Concretely, since Banquet-IOP is specialized for the relation $R = \left\{((\mathsf{ct}, \mathsf{pt}), K) \ : \ \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\right\}$, one can verifiably encrypt K satisfying the relation R with any PKE. To the best of our knowledge, no prior work proposed a verifiable encryption scheme for AES private keys. As AES keys are commonly stored in hardware, this is also relevant for our verifiable backup scenario. Since AES is considered PQ-secure, and encrypted data may have a long lifetime, in some systems it is important that AES keys be exported with a matching level of security. If PKE is instantiated with a quantum-resilient scheme, such as a lattice-based one, our verifiable encryption has PQ security, in the sense that both the *encryption scheme* and *relation* to be proven about the plaintext may withstand quantum attacks.

Verifiably encrypting an AES key with the hashed Elgamal scheme described in [TZ21] has proof sizes that are only slightly larger than Banquet proofs for AES, since the ciphertexts are only 16 bytes larger than hash-based commitments. For example, proofs are 20.4 KB ($N = 16, \tau = 41$), an overhead of less than 1KB, and ciphertexts are 2 KB. Prover and verifier times are dominated by the cost of computing encryptions, but we estimate the total time to be below 100ms (based on the time estimates used above and those from [BdK+21a]).

As we analyze in [TZ21], variants of the FO transform can be used for achieving undeniability and thus many efficient post-quantum PKE schemes, including Kyber [SAB+20] and FrodoKEM [NAB+19], are compatible with our framework. When our transform is applied to Banquet with Kyber-512, the proof and ciphertext sizes expand significantly (relative to Banquet proofs for AES), but remain practical, for example proofs are about 50 KB and ciphertexts 32.1 KB with the ($N = 16, \tau = 41$) parameters. The prover and verifier times are estimated to be below 200ms, roughly double the time required with Elgamal.

5.6 Conclusion and Future Work

As our construction gives a practical way to verifiably encrypt ECC, DSA, DH, RSA and AES keys, we have a complete and flexible solution to the verifiable backup problem for the most common key types stored in hardware and cloud services. A notable exception are keys for the HMAC algorithm. They can be handled with our transform and ZKB++ or KKW, but with larger proof sizes due to the larger circuit size of the SHA2 or SHA3 hash function. A more recent MPCitH protocol, Limbo [dOT21], can create proofs for SHA-256 that are 100-200 KB in size, and since Limbo is already described as an IOP it is a natural

candidate for our VE transform. For even larger circuits, an ideal approach would be a generalization of our compiler to construct VE schemes from more general IOPs, in order to make use of proof systems where communication is sub-linear in the circuit size (such as Ligero [AHIV17, BFH+20] and Aurora [BCR+19]), which currently outperform MPCitH proofs for large circuits. Another minor gap in our solution to the the verifiable backup problem is that strong validity (Section 5.3.5) is only guaranteed for perfectly correct PKE, which means we require a stronger assumption when PKE is lattice-based.

Related to encryption, can the ciphertexts produced by our construction be made CCA-secure? Currently they are if the entire proof transcript is sent to the receiver, however, once compression outputs a ciphertext, note that the ciphertext can be modified by dropping one of the individual PKE ciphertexts from one parallel repetition (even if PKE is CCA secure). Having PKE support labels (as discussed in [CS03]) might allow the set of PKE ciphertexts to be bound together. Also on the subject of CCA security, does CCA security imply undeniability?

The DKG-in-the-head design strategy proved useful here, and may be worth exploring further, since there is a large literature on distributed (or threshold) key generation upon which to draw inspiration. It is also an interesting open question whether our approach to VE leads to interesting instantiations of group and ring signatures, especially those targeting post-quantum security as was done in [BDK+21b], or those based only on symmetric-key primitives such as [KKW18, DRS18]. Finally, the performance and flexibility of our compression method to achieve constant-sized ciphertexts (discussed in Section 5.4.2) needs to be improved before it can be considered practical.

Bibliography

[A+] D. F. Aranha et al. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic. 20, 27

[AABN02] 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 *EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 418–433. Springer, Heidelberg, 2002. DOI: 10.1007/3-540-46035-7_28. 4, 8, 9, 49

[AASA+19] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. 2019. 41

[AB21] H. K. Alper and J. Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In *CRYPTO 2021, Part I*, vol. 12825 of *LNCS*, pp. 157–188, Virtual Event, 2021. Springer, Heidelberg. DOI: 10.1007/978-3-030-84242-0_7. 12, 68, 70

[ABC+22] D. F. Aranha, E. M. Bennedsen, M. Campanelli, C. Ganesh, C. Orlandi, and A. Takahashi. ECLIPSE: Enhanced Compiling Method for Pedersen-Committed zkSNARK Engines. In *PKC 2022*, vol. 13177 of *LNCS*, pp. 584–614. Springer, 2022. DOI: 10.1007/978-3-030-97121-2_21, Full version available at https://eprint.iacr.org/2021/934.pdf. 13, 15

[ABE+21] D. F. Aranha, S. Berndt, T. Eisenbarth, O. Seker, A. Takahashi, L. Wilke, and G. Zaverucha. Side-channel protections for Picnic signatures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):239–282, 2021. DOI: 10.46586/tches.v2021.i4.239-282, Full version available at https://eprint.iacr.org/2021/735.pdf. Preliminary version appeared at the 3rd NIST PQC Standardization Conference. 15

[ABF+16] T. Allan, B. B. Brumley, K. E. Falkner, J. van de Pol, and Y. Yarom. Amplifying side channels through performance degradation. In *ACSAC*, pp. 422–435, 2016. 25, 29

[ABF+18] C. Ambrose, J. W. Bos, B. Fay, M. Joye, M. Lochter, and B. Murray. Differential attacks on deterministic signatures. In *CT-RSA 2018*, vol. 10808 of *LNCS*, pp. 339–353. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-76953-0_18. 11, 41, 43, 49

[ABG10] O. Aciiçmez, B. B. Brumley, and P. Grabher. New results on instruction cache attacks. In *CHES 2010*, vol. 6225 of *LNCS*, pp. 110–124. Springer, Heidelberg, 2010. DOI: 10.1007/978-3-642-15031-9_8. 22

[ABuH+19] A. C. Aldaya, B. B. Brumley, S. ul Hassan, C. P. García, and N. Tuveri. Port contention for fun and profit. In *2019 IEEE Symposium on Security and Privacy*, pp. 870–887. IEEE Computer Society Press, 2019. DOI: 10.1109/SP.2019.00066. 10, 22, 25

[ACM+17] P. Austrin, K. Chung, M. Mahmoody, R. Pass, and K. Seth. On the impossibility of cryptography with tamperable randomness. *Algorithmica*, 79(4):1052–1101, 2017. DOI: 10.1007/s00453-016-0219-7, https://doi.org/10.1007/s00453-016-0219-7. 45

[AF04] M. Abe and S. Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In *CRYPTO 2004*, vol. 3152 of *LNCS*, pp. 317–334. Springer, Heidelberg, 2004. DOI: 10.1007/978-3-540-28628-8_20. 68

[AFG+14] D. F. Aranha, P.-A. Fouque, B. Gérard, J.-G. Kammerer, M. Tibouchi, and J.-C. Zapalowicz. GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In *ASIACRYPT 2014, Part I*, vol. 8873 of *LNCS*, pp. 262–281. Springer, Heidelberg, 2014. DOI: 10.1007/978-3-662-45611-8_14. 19, 20, 21, 25, 26, 30, 35, 37, 38, 51, 60

[AFLT16] M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, 2016. DOI: 10.1007/s00145-015-9203-7. 6, 73, 75

[AGM18] S. Agrawal, C. Ganesh, and P. Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *CRYPTO 2018, Part III*, vol. 10993 of *LNCS*, pp. 643–673. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-96878-0_22. 13

[AGS07] O. Aciiçmez, S. Gueron, and J.-P. Seifert. New branch prediction vulnerabilities in OpenSSL and necessary software countermeasures. In *11th IMA International Conference on Cryptography and Coding*, vol. 4887 of *LNCS*, pp. 185–203. Springer, Heidelberg, 2007. 22

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 2017*, pp. 2087–2104. ACM Press, 2017. DOI: 10.1145/3133956.3134104. 111, 132

[AKV22] Microsoft Azure Key Vault Documentation: Key types, algorithms, and operations, https://docs.microsoft.com/en-us/azure/key-vault/keys/about-keys-details. 105, 106

[ANT+20a] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In *ACM CCS 2020*, pp. 225–242. ACM Press, 2020. DOI: 10.1145/3372297.3417268. 10

[ANT+20b] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. Cryptology ePrint Archive, Report 2020/615, Full version. Available at https://eprint.iacr.org/2020/615.pdf. 10, 21, 23, 34, 37, 38, 39, 40

[AOTZ19] D. F. Aranha, C. Orlandi, A. Takahashi, and G. Zaverucha. Security of hedged Fiat-Shamir signatures under fault attacks. Cryptology ePrint Archive, Report 2019/956, Full version. Available at https://eprint.iacr.org/2019/956.pdf. 11, 46, 47, 49, 51, 62, 64

[AOTZ20] D. F. Aranha, C. Orlandi, A. Takahashi, and G. Zaverucha. Security of hedged Fiat-Shamir signatures under fault attacks. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 644–674. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45721-1_23. 11

[AS07] O. Acıiçmez and J.-P. Seifert. Cheap hardware parallelism implies cheap security. In *Fourth International Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 80–91, Vienna, AT, 2007. 22

[ASW98] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT'98*, vol. 1403 of *LNCS*, pp. 591–606. Springer, Heidelberg, 1998. DOI: 10.1007/BFb0054156. 106

[Ate99] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *ACM CCS 99*, pp. 138–146. ACM Press, 1999. DOI: 10.1145/319709.319728. 106

[AWS22a] Amazon Web Services CloudHSM Documentation: Using the command line to manage keys, https://docs.aws.amazon.com/cloudhsm/latest/userguide/using-kmu.html. 105, 106

[AWS22b] Amazon Web Services Key Management Service Documentation: AWS KMS Keys, https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#kms_keys. 105

[BAA+19] N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon. qTESLA. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 11, 41, 68, 74

[Bab85] L. Babai. Trading group theory for randomness. In *17th ACM STOC*, pp. 421–429. ACM Press, 1985. DOI: 10.1145/22145.22192. 1

[Bae14] M. Baert. Ed25519 leaks private key if public key is incorrect #170. https://github.com/jedisct1/libsodium/issues/170, 2014. 11, 41

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pp. 315–334. IEEE Computer Society Press, 2018. DOI: 10.1109/SP.2018.00020. 106, 110

[BBE+18] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi. Masking the GLP lattice-based signature scheme at any order. In *EUROCRYPT 2018, Part II*, vol. 10821 of *LNCS*, pp. 354–384. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-78375-8_12. 69, 74, 78

[BBE+19] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, M. Rossi, and M. Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In *ACM CCS 2019*, pp. 2147–2164. ACM Press, 2019. DOI: 10.1145/3319535.3363223. 69

[BBG+17] D. J. Bernstein, J. Breitner, D. Genkin, L. G. Bruinderink, N. Heninger, T. Lange, C. van Vredendaal, and Y. Yarom. Sliding right into disaster: Left-to-right sliding windows leak. In *CHES 2017*, vol. 10529 of *LNCS*, pp. 555–576. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-66787-4_27. 22

[BBN+09] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 232–249. Springer, Heidelberg, 2009. DOI: 10.1007/978-3-642-10366-7_14. 44

[BCJ08] A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *ACM CCS 2008*, pp. 449–458. ACM Press, 2008. DOI: 10.1145/1455770.1455827. 68, 70, 86, 100

[BCJ11] A. Becker, J.-S. Coron, and A. Joux. Improved generic algorithms for hard knapsacks. In *EUROCRYPT 2011*, vol. 6632 of *LNCS*, pp. 364–385. Springer, Heidelberg, 2011. DOI: 10.1007/978-3-642-20465-4_21. 34

[BCK+14] F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT 2014, Part I*, vol. 8873 of *LNCS*, pp. 551–572. Springer, Heidelberg, 2014. DOI: 10.1007/978-3-662-45611-8_29. 69, 78

[BCN+06] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006. DOI: 10.1109/JPROC.2005.862424, https://doi.org/10.1109/JPROC.2005.862424. 45

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, pp. 103–128. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-17653-2_4. 111, 132

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, vol. 9986 of *LNCS*, pp. 31–60. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53644-5_2. 108, 111

[Bd20] W. Beullens and C. de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *Post-Quantum Cryptography - 11th*

International Conference, PQCrypto 2020, pp. 130–150. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-44223-1_8. 107

[BD21] M. Bellare and W. Dai. Chain reductions for multi-signatures and the HBMS scheme. In *ASIACRYPT 2021, Part IV*, vol. 13093 of *LNCS*, pp. 650–678. Springer, Heidelberg, 2021. DOI: 10.1007/978-3-030-92068-5_22. 12, 68

[BDD20] C. Baum, B. David, and R. Dowsley. (Public) Verifiability for composable protocols without adaptivity or zero-knowledge. Cryptology ePrint Archive, Report 2020/207, https://ia.cr/2020/207. 112

[BDF+11] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 41–69. Springer, Heidelberg, 2011. DOI: 10.1007/978-3-642-25385-0_3. 8

[BDF+14] G. Barthe, F. Dupressoir, P.-A. Fouque, B. Grégoire, M. Tibouchi, and J.-C. Zapalowicz. Making RSA-PSS provably secure against non-random faults. In *CHES 2014*, vol. 8731 of *LNCS*, pp. 206–222. Springer, Heidelberg, 2014. DOI: 10.1007/978-3-662-44709-3_12. 44, 54

[BDG+13] N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. T. Kalai, A. López-Alt, and D. Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In *TCC 2013*, vol. 7785 of *LNCS*, pp. 182–201. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-36594-2_11. 8

[BDG20] M. Bellare, H. Davis, and F. Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In *EUROCRYPT 2020, Part II*, vol. 12106 of *LNCS*, pp. 3–32. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45724-2_1. 86

[BdK+21a] C. Baum, C. de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha. Banquet: Short and fast signatures from AES. In *PKC 2021, Part I*, vol. 12710 of *LNCS*, pp. 266–297. Springer, Heidelberg, 2021. DOI: 10.1007/978-3-030-75245-3_11. 107, 108, 109, 122, 131

[BDK+21b] W. Beullens, S. Dobson, S. Katsumata, Y.-F. Lai, and F. Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. Cryptology ePrint Archive, Report 2021/1366, https://eprint.iacr.org/2021/1366. 111, 132

[BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT'97*, vol. 1233 of *LNCS*, pp. 37–51. Springer, Heidelberg, 1997. DOI: 10.1007/3-540-69053-0_4. 10, 44

[BDL+12] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012. DOI: 10.1007/s13389-012-0027-1. 11, 40, 41

[BDL+18] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In *SCN 18*, vol. 11035 of *LNCS*, pp. 368–385. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-98113-0_20. 69, 100, 101, 104

[Beu20] W. Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In *EUROCRYPT 2020, Part III*, vol. 12107 of *LNCS*, pp. 183–211. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45727-3_7. 107, 111

[BFH+20] R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang. Ligero++: A new optimized sublinear IOP. In *ACM CCS 2020*, pp. 2025–2038. ACM Press, 2020. DOI: 10.1145/3372297.3417893. 115, 132

[BFMT16] P. Belgarric, P.-A. Fouque, G. Macario-Rat, and M. Tibouchi. Side-channel analysis of Weierstrass and Koblitz curve ECDSA on android smartphones. In *CT-RSA 2016*, vol. 9610 of *LNCS*, pp. 236–252. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-319-29485-8_14. 25

[BG93] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO'92*, vol. 740 of *LNCS*, pp. 390–420. Springer, Heidelberg, 1993. DOI: 10.1007/3-540-48071-4_28. 3

[BGG+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO 2018, Part I*, vol. 10991 of *LNCS*, pp. 565–596. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-96884-1_19. 67, 74, 75

[BH15] M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, pp. 627–656. Springer, Heidelberg, 2015. DOI: 10.1007/978-3-662-46803-6_21. 44

[BH19] J. Breitner and N. Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In *FC 2019*, vol. 11598 of *LNCS*, pp. 3–20. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-32101-7_1. 10

[BHH+19] M. Backes, L. Hanzlik, A. Herzberg, A. Kate, and I. Pryvalov. Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In *PKC 2019, Part I*, vol. 11442 of *LNCS*, pp. 286–313. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-17253-4_10. 13

[BHLY16] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In *CHES 2016*, vol. 9813 of *LNCS*, pp. 323–345. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53140-2_16. 22

[BK03] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *EUROCRYPT 2003*, vol. 2656 of *LNCS*, pp. 491–506. Springer, Heidelberg, 2003. DOI: 10.1007/3-540-39200-9_31. 45

[BKLP15] F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS 2015, Part I*, vol. 9326 of *LNCS*, pp. 305–325. Springer, Heidelberg, 2015. DOI: 10.1007/978-3-319-24174-6_16. 74

[BKP13] R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *ACNS 13*, vol. 7954 of *LNCS*, pp. 218–236. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-38980-1_14. 67, 74, 75

[BL07] D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In *ASIACRYPT 2007*, vol. 4833 of *LNCS*, pp. 29–50. Springer, Heidelberg, 2007. DOI: 10.1007/978-3-540-76900-2_3. 23

[Ble00] D. Bleichenbacher. On the generation of one-time keys in DL signature schemes. Presentation at IEEE P1363 working group meeting, 2000. 10, 11, 19, 20, 25, 60

[Ble05] D. Bleichenbacher. Experiments with DSA. Rump session at CRYPTO 2005, available from https://www.iacr.org/conferences/crypto2005/r/3.pdf. 11, 19, 20, 37

[BLS19] J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 176–202. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26948-7_7. 74

[Blu86] M. Blum. How to prove a theorem so no one can claim it. In *Proceedings of the International Congress of Mathematicians*, pp. 1444–1451, 1986. 1

[BN06] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pp. 390–399. ACM Press, 2006. DOI: 10.1145/1180405.1180453. 68, 69, 73, 75, 82, 84, 90, 100

[BN20] C. Baum and A. Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In *PKC 2020, Part I*, vol. 12110 of *LNCS*, pp. 495–526. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45374-9_17. 107, 111

[BP02] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 162–177. Springer, Heidelberg, 2002. DOI: 10.1007/3-540-45708-9_11. 5

[BP16] A. Barenghi and G. Pelosi. A note on fault attacks against deterministic signature schemes. In *IWSEC 16*, vol. 9836 of *LNCS*, pp. 182–192. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-319-44524-3_11. 11, 41, 43, 49

[BP18] L. G. Bruinderink and P. Pessl. Differential fault attacks on deterministic lattice signatures. *IACR TCHES*, 2018(3):21–43, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7267. 11, 41, 42

[BPS16] M. Bellare, B. Poettering, and D. Stebila. From identification to signatures, tightly: A framework and generic transforms. In *ASIACRYPT 2016, Part II*, vol. 10032 of *LNCS*, pp. 435–464. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53890-6_15. 43, 44, 50, 51, 56

[BPS17] A. Boldyreva, C. Patton, and T. Shrimpton. Hedging public-key encryption in the real world. In *CRYPTO 2017, Part III*, vol. 10403 of *LNCS*, pp. 462–494. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-63697-9_16. 44

[BPW12] D. Bernhard, O. Pereira, and B. Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In *ASIACRYPT 2012*, vol. 7658 of *LNCS*, pp. 626–643. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-34961-4_38. 6

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pp. 62–73. ACM Press, 1993. DOI: 10.1145/168588.168596. 8

[BR18] M. Brengel and C. Rossow. Identifying key leakage of bitcoin users. In *RAID*, vol. 11050 of *Lecture Notes in Computer Science*, pp. 623–643. Springer, 2018. 41

[BR21] K. Boudgoust and A. Roux-Langlois. Compressed linear aggregate signatures based on module lattices. Cryptology ePrint Archive, Report 2021/263, https://eprint.iacr.org/2021/263. 74

[BS13] S. Bettaieb and J. Schrek. Improved lattice-based threshold ring signature scheme. In *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pp. 34–51. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-38616-9_3. 74

[BT11] B. B. Brumley and N. Tuveri. Remote timing attacks are still practical. In *ESORICS 2011*, vol. 6879 of *LNCS*, pp. 355–371. Springer, Heidelberg, 2011. DOI: 10.1007/978-3-642-23822-2_20. 25, 29

[BT16] M. Bellare and B. Tackmann. Nonce-based cryptography: Retaining security when randomness fails. In *EUROCRYPT 2016, Part I*, vol. 9665 of *LNCS*, pp. 729–757. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-49890-3_28. 43, 44, 50, 51

[BTT22] C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *CRYPTO 2022*, LNCS. Springer, To appear. 15, 16

[BV96] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In *CRYPTO'96*,

vol. 1109 of *LNCS*, pp. 129–142. Springer, Heidelberg, 1996. DOI: 10.1007/3-540-68697-5_11. 11, 19, 24, 25, 60

[BvSY14] N. Benger, J. van de Pol, N. P. Smart, and Y. Yarom. "ooh aah... just a little bit": A small amount of side channel can go a long way. In *CHES 2014*, vol. 8731 of *LNCS*, pp. 75–92. Springer, Heidelberg, 2014. DOI: 10.1007/978-3-662-44709-3_5. 25

[CAGB20] A. Cabrera Aldaya, C. P. García, and B. B. Brumley. From A to Z: projective coordinates leakage in the wild. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):428–453, 2020. DOI: 10.13154/tches.v2020.i3.428-453, https://doi.org/10.13154/tches.v2020.i3.428-453. 40

[CCD+17] K. Cohn-Gordon, C. J. F. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *EuroS&P*, pp. 451–466. IEEE, 2017. 42

[CCFG16] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In *ACM CCS 2016*, pp. 1614–1625. ACM Press, 2016. DOI: 10.1145/2976749.2978337. 106

[CCH+19] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, and D. Wichs. Fiat-Shamir: from practice to theory. In *51st ACM STOC*, pp. 1082–1090. ACM Press, 2019. DOI: 10.1145/3313276.3316380. 8

[CCL+19] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO 2019, Part III*, vol. 11694 of *LNCS*, pp. 191–221. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26954-8_7. 67

[CCL+20] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC 2020, Part II*, vol. 12111 of *LNCS*, pp. 266–296. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45388-6_10. 67

[CCRR18] R. Canetti, Y. Chen, L. Reyzin, and R. D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 91–122. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-78381-9_4. 8

[CD00] J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 331–345. Springer, Heidelberg, 2000. DOI: 10.1007/3-540-44448-3_25. 13, 14, 15, 105, 106, 109, 110, 112, 114, 128, 130

[CDG+17] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS 2017*, pp. 1825–1842. ACM Press, 2017. DOI: 10.1145/3133956.3133997. 108, 121

[CDN15] R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015, http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053. 2

[CDNO97] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *CRYPTO'97*, vol. 1294 of *LNCS*, pp. 90–104. Springer, Heidelberg, 1997. DOI: 10.1007/BFb0052229. 112

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, vol. 839 of *LNCS*, pp. 174–187. Springer, Heidelberg, 1994. DOI: 10.1007/3-540-48658-5_19. 3, 6

[CF85] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th FOCS*, pp. 372–382. IEEE Computer Society Press, 1985. DOI: 10.1109/SFCS.1985.2. 106

[CFF+20] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. Cryptology ePrint Archive, Report 2020/1069, https://eprint.iacr.org/2020/1069. 115

[CFF+21] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: A toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *ASIACRYPT 2021*, vol. 13092 of *LNCS*, pp. 3–33. Springer, 2021. DOI: 10.1007/978-3-030-92078-4_1, https://doi.org/10.1007/978-3-030-92078-4_1. 13

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multiparty computation. In *28th ACM STOC*, pp. 639–648. ACM Press, 1996. DOI: 10.1145/237814.238015. 112

[CFQ19] M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In *ACM CCS 2019*, pp. 2075–2092. ACM Press, 2019. DOI: 10.1145/3319535.3339820. 13, 110

[CGG+20] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *ACM CCS 2020*, pp. 1769–1787. ACM Press, 2020. DOI: 10.1145/3372297.3423367. 67

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pp. 209–218. ACM Press, 1998. DOI: 10.1145/276698.276741. 8

[CGM16] M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *CRYPTO 2016, Part III*, vol. 9816 of *LNCS*, pp. 499–530. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53015-3_18. 13

[Cha19] A. Chailloux. Quantum security of the Fiat-Shamir transform of commit and open protocols. Cryptology ePrint Archive, Report 2019/699, https://eprint.iacr.org/2019/699. 48

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 523–552. Springer, Heidelberg, 2010. DOI: 10.1007/978-3-642-13190-5_27. 70, 74

[CHM+20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 738–768. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45721-1_26. 115

[CK16] R. Choi and K. Kim. Lattice-based multi-signature with linear homomorphism. In *2016 Symposium on Cryptography and Information Security (SCIS 2016)*, 2016. 74

[CL06] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *CRYPTO 2006*, vol. 4117 of *LNCS*, pp. 78–96. Springer, Heidelberg, 2006. DOI: 10.1007/11818175_5. 112

[Cla07] C. Clavier. Secret external encodings do not prevent transient fault analysis. In *CHES 2007*, vol. 4727 of *LNCS*, pp. 181–194. Springer, Heidelberg, 2007. DOI: 10.1007/978-3-540-74735-2_13. 45, 51

[CLRS10] P. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. In *LATINCRYPT 2010*, vol. 6212 of *LNCS*, pp. 255–272. Springer, 2010. DOI: 10.1007/978-3-642-14712-8_16, https://doi.org/10.1007/978-3-642-14712-8_16. 74

[CM09] J.-S. Coron and A. Mandal. PSS is secure against random fault attacks. In *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 653–666. Springer, Heidelberg, 2009. DOI: 10.1007/978-3-642-10366-7_38. 44, 54

[Cor99] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES'99*, vol. 1717 of *LNCS*, pp. 292–302. Springer, Heidelberg, 1999. DOI: 10.1007/3-540-48059-5_25. 40

[COSV17a] M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In *TCC 2017, Part I*, vol. 10677 of *LNCS*, pp. 711–742. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-70500-2_24. 74

[COSV17b] M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 127–157. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-63715-0_5. 74

[CPS+16a] M. Ciampi, G. Persiano, A. Scafuro, L. Siniscalchi, and I. Visconti. Improved OR-composition of sigma-protocols. In *TCC 2016-A, Part II*, vol.

9563 of *LNCS*, pp. 112–141. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-49099-0_5. 6, 55, 60, 74

[CPS+16b] M. Ciampi, G. Persiano, A. Scafuro, L. Siniscalchi, and I. Visconti. Online/offline OR composition of sigma protocols. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 63–92. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-49896-5_3. 6

[Cra96] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI, Amsterdam, https://ir.cwi.nl/pub/21438. 3

[CS03] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 126–144. Springer, Heidelberg, 2003. DOI: 10.1007/978-3-540-45146-4_8. 13, 106, 110, 128, 132

[CS18] C. Costello and B. Smith. Montgomery curves and their arithmetic - the case of large characteristic fields. *Journal of Cryptographic Engineering*, 8(3):227–240, 2018. DOI: 10.1007/s13389-017-0157-6. 23

[CS19] D. Cozzo and N. P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In *17th IMA International Conference on Cryptography and Coding*, vol. 11929 of *LNCS*, pp. 128–153. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-35199-1_7. 68

[Dam00] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 418–430. Springer, Heidelberg, 2000. DOI: 10.1007/3-540-45539-6_30. 73, 74

[Dam10] I. Damgård. On Σ-protocols. Lectures notes from Cryptologic Protocol Theory, https://cs.au.dk/~ivan/Sigma.pdf. 3, 49

[DAN+18] L. De Meyer, V. Arribas, S. Nikova, V. Nikov, and V. Rijmen. M&M: Masks and macs against physical attacks. *IACR TCHES*, 2019(1):25–50, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7333. 45

[DDE+18] F. Dall, G. De Micheli, T. Eisenbarth, D. Genkin, N. Heninger, A. Moghimi, and Y. Yarom. CacheQuote: Efficiently recovering long-term secrets of SGX EPID via cache attacks. *IACR TCHES*, 2018(2):171–191, 2018. https://tches.iacr.org/index.php/TCHES/article/view/879. 10, 25

[DDE+20] J. Daemen, C. Dobraunig, M. Eichlseder, H. Gross, F. Mendel, and R. Primas. Protecting against statistical ineffective fault attacks. *IACR TCHES*, 2020(3):508–543, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8599. 45

[DDLL13] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 40–56. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-40041-4_3. 74

[dDOS19] C. de Saint Guilhem, L. De Meyer, E. Orsini, and N. P. Smart. BBQ: Using AES in picnic signatures. In *SAC 2019*, vol. 11959 of *LNCS*, pp. 669–692. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-38471-5_27. 107

[DEF+19] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pp. 1084–1101. IEEE Computer Society Press, 2019. DOI: 10.1109/SP.2019.00050. 12, 68, 70, 86, 90, 92

[DEG+18] C. Dobraunig, M. Eichlseder, H. Groß, S. Mangard, F. Mendel, and R. Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In *ASIACRYPT 2018, Part II*, vol. 11273 of *LNCS*, pp. 315–342. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-030-03329-3_11. 45

[DEK+18] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR TCHES*, 2018(3):547–572, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7286. 45, 51

[DFG13] Ö. Dagdelen, M. Fischlin, and T. Gagliardoni. The Fiat-Shamir transformation in a quantum world. In *ASIACRYPT 2013, Part II*, vol. 8270 of *LNCS*, pp. 62–81. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-42045-0_4. 8

[DFM20] J. Don, S. Fehr, and C. Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In *CRYPTO 2020, Part III*, vol. 12172 of *LNCS*, pp. 602–631. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-56877-1_21. 8

[DFMS19] J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In *CRYPTO 2019, Part II*, vol. 11693 of *LNCS*, pp. 356–383. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26951-7_13. 8

[DFMS21] J. Don, S. Fehr, C. Majenz, and C. Schaffner. Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280, https://eprint.iacr.org/2021/280. 8, 112, 119

[DFMS22] J. Don, S. Fehr, C. Majenz, and C. Schaffner. Efficient nizks and signatures from commit-and-open protocols in the qrom. Cryptology ePrint Archive, Report 2022/270, https://ia.cr/2022/270. 8

[DFMV17] I. Damgård, S. Faust, P. Mukherjee, and D. Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. *Journal of Cryptology*, 30(1):152–190, 2017. DOI: 10.1007/s00145-015-9218-0. 45, 54

[DGRW18] Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage. Fast message franking: From invisible salamanders to encryptment. In *CRYPTO 2018, Part I*, vol. 10991 of *LNCS*, pp. 155–186. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-96884-1_6. 112

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. DOI: 10.1109/TIT.1976.1055638, https://doi.org/10.1109/TIT.1976.1055638. 7

[DHH+15] M. Düll, B. Haase, G. Hinterwälder, M. Hutter, C. Paar, A. H. Sánchez, and P. Schwabe. High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. *Des. Codes Cryptogr.*, 77(2-3):493–514, 2015. 40

[DHMP13] E. De Mulder, M. Hutter, M. E. Marson, and P. Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In *CHES 2013*, vol. 8086 of *LNCS*, pp. 435–452. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-40349-1_25. 19

[DHMP14] E. De Mulder, M. Hutter, M. E. Marson, and P. Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. *Journal of Cryptographic Engineering*, 4(1):33–45, 2014. DOI: 10.1007/s13389-014-0072-z. 25, 26, 34, 35, 39

[DHSS20] Y. Doröz, J. Hoffstein, J. H. Silverman, and B. Sunar. Mmsat: A scheme for multimessage multiuser signature aggregation. Cryptology ePrint Archive, Report 2020/520, https://eprint.iacr.org/2020/520. 74

[Din19] I. Dinur. An algorithmic framework for the generalized birthday problem. *Des. Codes Cryptogr.*, 87(8):1897–1926, 2019. DOI: 10.1007/s10623-018-00594-6, https://doi.org/10.1007/s10623-018-00594-6. 27, 31, 34

[DJN+20] I. Damgård, T. P. Jakobsen, J. B. Nielsen, J. I. Pagter, and M. B. Østergaard. Fast threshold ECDSA with honest majority. In *SCN 20*, vol. 12238 of *LNCS*, pp. 382–400. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-57990-6_19. 67

[DKLs18] J. Doerner, Y. Kondi, E. Lee, and a. shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pp. 980–997. IEEE Computer Society Press, 2018. DOI: 10.1109/SP.2018.00036. 67

[DKLs19] J. Doerner, Y. Kondi, E. Lee, and a. shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pp. 1051–1066. IEEE Computer Society Press, 2019. DOI: 10.1109/SP.2019.00024. 67

[DLL+18] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals–dilithium: Digital signatures from module lattices. 2018, https://repository.ubn.ru.nl/bitstream/handle/2066/191703/191703.pdf. 69, 73, 76, 85, 87

[dLS18] R. del Pino, V. Lyubashevsky, and G. Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM CCS 2018*, pp. 574–591. ACM Press, 2018. DOI: 10.1145/3243734.3243852. 74

[DM14] L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014, Part I*, vol. 8616 of *LNCS*, pp. 335–352. Springer, Heidelberg, 2014. DOI: 10.1007/978-3-662-44371-2_19. 70, 74

[DN00] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 432–450. Springer, Heidelberg, 2000. DOI: 10.1007/3-540-44598-6_27. 112

[DOK+20] A. P. K. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS 2020, Part II*, vol. 12309 of *LNCS*, pp. 654–673. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-59013-0_32. 67

[dOT21] C. de Saint Guilhem, E. Orsini, and T. Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In *ACM CCS 2021*, pp. 3022–3036. ACM Press, 2021. DOI: 10.1145/3460120.3484595. 107, 111, 115, 131

[DOTT20] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. Cryptology ePrint Archive, Report 2020/1110, Full version. Available at https://eprint.iacr.org/2020/1110.pdf. 13, 69, 72, 73, 86, 100

[DOTT21] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC 2021, Part I*, vol. 12710 of *LNCS*, pp. 99–130. Springer, Heidelberg, 2021. DOI: 10.1007/978-3-030-75245-3_5. 13

[DOTT22] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *J. Cryptol.*, 35(14), 2022. DOI: 10.1007/s00145-022-09425-3. 13

[DPW18] S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018. DOI: 10.1145/3178432, https://doi.org/10.1145/3178432. 45

[DRS18] D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pp. 419–440. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-79063-3_20. 132

[EEE20] M. F. Esgin, O. Ersoy, and Z. Erkin. Post-quantum adaptor signatures and payment channel networks. In *ESORICS 2020, Part II*, vol. 12309 of *LNCS*, pp. 378–397. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-59013-0_19. 74

[EFG+22] T. Espitau, P. Fouque, F. Gérard, M. Rossi, A. Takahashi, M. Tibouchi, A. Wallet, and Y. Yu. Mitaka: A simpler, parallelizable, maskable variant of Falcon. In *EUROCRYPT 2022*, vol. 13277 of *LNCS*, pp. 222–253.

Springer, 2022. DOI: 10.1007/978-3-031-07082-2_9, Full version available at https://eprint.iacr.org/2021/1486.pdf. Preliminary version appeared at the 3rd NIST PQC Standardization Conference. 15, 16

[EG21] Electionguard specification, available at https://www.electionguard.vote/. 106

[ES16] R. El Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In *CANS 16*, vol. 10052 of *LNCS*, pp. 140–155. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-319-48965-0_9. 68, 69, 73, 74

[ESLL19] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 115–146. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26948-7_5. 74, 77

[ESS+19] M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS 19*, vol. 11464 of *LNCS*, pp. 67–88. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-21568-2_4. 69, 74

[fai10] fail0verflow. Console hacking 2010 – PS3 epic fail. 27th Chaos Communications Congress, 2010. 41

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. DOI: 10.1007/BF02351717. 3, 4

[FG20] M. Fischlin and F. Günther. Modeling memory faults in signature and authenticated encryption schemes. In *CT-RSA 2020*, vol. 12006 of *LNCS*, pp. 56–84. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-40186-3_4. 45

[FGMN16] P. Fouque, S. Guilley, C. Murdica, and D. Naccache. Safe-errors on SPA protected implementations with the atomicity technique. In *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, vol. 9100 of *Lecture Notes in Computer Science*, pp. 479–493. Springer, 2016. DOI: 10.1007/978-3-662-49301-4_30, https://doi.org/10.1007/978-3-662-49301-4_30. 31

[FH19] M. Fukumitsu and S. Hasegawa. A tightly-secure lattice-based multisignature. In *APKC@AsiaCCS 2019*, pp. 3–11. ACM, 2019. DOI: 10.1145/3327958.3329542, https://doi.org/10.1145/3327958.3329542. 68, 69, 74

[FH20] M. Fukumitsu and S. Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In *ProvSec 2020*, vol. 12505 of *LNCS*, pp. 45–64. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-62576-4_3. 68, 69, 74, 75, 78

[FHJ20] M. Fischlin, P. Harasser, and C. Janson. Signatures from sequential-OR proofs. In *EUROCRYPT 2020, Part III*, vol. 12107 of *LNCS*, pp. 212–244. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-45727-3_8. 6

[Fil11] J. Fildes. iPhone hacker publishes secret Sony PlayStation 3 key. BBC News, accessed on April 10, 2022. Available at https://www.bbc.com/news/technology-12116051. 10

[Fis05] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, vol. 3621 of *LNCS*, pp. 152–168. Springer, Heidelberg, 2005. DOI: 10.1007/11535218_10. 8, 112

[FJ05] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, special issue on "Program Generation, Optimization, and Platform Adaptation". 38

[FKMV12] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT 2012*, vol. 7668 of *LNCS*, pp. 60–79. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-34931-7_5. 8

[FNSV18] A. Faonio, J. B. Nielsen, M. Simkin, and D. Venturi. Continuously non-malleable codes with split-state refresh. In *ACNS 18*, vol. 10892 of *LNCS*, pp. 121–139. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-93387-0_7. 45

[FO99] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC'99*, vol. 1560 of *LNCS*, pp. 53–68. Springer, Heidelberg, 1999. DOI: 10.1007/3-540-49162-7_5. 108, 119

[FO13] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, 2013. DOI: 10.1007/s00145-011-9114-1. 108, 119

[FPV11] S. Faust, K. Pietrzak, and D. Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *ICALP 2011, Part I*, vol. 6755 of *LNCS*, pp. 391–402. Springer, Heidelberg, 2011. DOI: 10.1007/978-3-642-22006-7_33. 44

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 186–194. Springer, Heidelberg, 1987. DOI: 10.1007/3-540-47721-7_12. 4, 6, 42, 108, 109, 122, 128

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pp. 416–426. ACM Press, 1990. DOI: 10.1145/100216.100272. 6

[FV16] A. Faonio and D. Venturi. Efficient public-key cryptography with bounded leakage and tamper resilience. In *ASIACRYPT 2016, Part I*, vol. 10031 of *LNCS*, pp. 877–907. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53887-6_32. 45

[FX16] E. Fujisaki and K. Xagawa. Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions. In *ASIACRYPT 2016, Part I*,

vol. 10031 of *LNCS*, pp. 908–938. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53887-6_33. 45

[GB17] C. P. García and B. B. Brumley. Constant-time callees with variable-time callers. In *USENIX Security 2017*, pp. 83–98. USENIX Association, 2017. 22, 25

[GBK11] D. Gullasch, E. Bangerter, and S. Krenn. Cache games - bringing access-based cache attacks on AES to practice. In *2011 IEEE Symposium on Security and Privacy*, pp. 490–505. IEEE Computer Society Press, 2011. DOI: 10.1109/SP.2011.22. 22

[GCZ16] S. Goldfeder, M. Chase, and G. Zaverucha. Efficient post-quantum zero-knowledge and signatures. Cryptology ePrint Archive, Report 2016/1110, https://eprint.iacr.org/2016/1110. 107

[GG18] R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *ACM CCS 2018*, pp. 1179–1194. ACM Press, 2018. DOI: 10.1145/3243734.3243859. 67

[GGN16] R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS 16*, vol. 9696 of *LNCS*, pp. 156–174. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-319-39555-5_9. 67

[GGSW13] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *45th ACM STOC*, pp. 467–476. ACM Press, 2013. DOI: 10.1145/2488608.2488667. 112

[GH03] Y. Gertner and A. Herzberg. Committing encryption and publicly-verifiable signcryption. Cryptology ePrint Archive, Report 2003/254, https://eprint.iacr.org/2003/254. 112, 113

[GHHM21] A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. Tight adaptive reprogramming in the QROM. 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 I*, vol. 13090 of *Lecture Notes in Computer Science*, pp. 637–667. Springer, 2021. DOI: 10.1007/978-3-030-92062-3_22, https://doi.org/10.1007/978-3-030-92062-3_22. 47

[GHM+21] K. Gjøsteen, T. Haines, J. Müller, P. Rønne, and T. Silde. Verifiable decryption in the head. Cryptology ePrint Archive, Report 2021/558, https://eprint.iacr.org/2021/558. 111

[GJKR07] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007. DOI: 10.1007/s00145-006-0347-3. 68

[GJKW07] E.-J. Goh, S. Jarecki, J. Katz, and N. Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, 2007. DOI: 10.1007/s00145-007-0549-3. 6

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996. DOI: 10.1137/S0097539791220688, https://doi.org/10.1137/S0097539791220688. 6

[GK03] S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pp. 102–115. IEEE Computer Society Press, 2003. DOI: 10.1109/SFCS.2003.1238185. 8

[GK22] Google Cloud Key Management Service Documentation: Key purposes and algorithms, https://cloud.google.com/kms/docs/algorithms. 105, 106

[GKMN21] F. Garillot, Y. Kondi, P. Mohassel, and V. Nikolaenko. Threshold Schnorr with stateless deterministic signing from standard assumptions. In *CRYPTO 2021, Part I*, vol. 12825 of *LNCS*, pp. 127–156, Virtual Event, 2021. Springer, Heidelberg. DOI: 10.1007/978-3-030-84242-0_6. 12

[GKSS20] A. Gagol, J. Kula, D. Straszak, and M. Swietek. Threshold ecdsa for decentralized asset custody. Cryptology ePrint Archive, Report 2020/498, https://eprint.iacr.org/2020/498. 67

[GLM+04] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC 2004*, vol. 2951 of *LNCS*, pp. 258–277. Springer, Heidelberg, 2004. DOI: 10.1007/978-3-540-24638-1_15. 44, 53

[GLP12] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES 2012*, vol. 7428 of *LNCS*, pp. 530–547. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-33027-8_31. 68, 74

[GLR17] P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. In *CRYPTO 2017, Part III*, vol. 10403 of *LNCS*, pp. 66–97. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-63697-9_3. 112

[GM18] N. Genise and D. Micciancio. Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 174–203. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-78381-9_7. 104

[GMO16] I. Giacomelli, J. Madsen, and C. Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In *USENIX Security 2016*, pp. 1069–1083. USENIX Association, 2016. 107, 108, 121

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pp. 291–304. ACM Press, 1985. DOI: 10.1145/22145.22178. 1, 3

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. DOI: 10.1137/0217017, https://doi.org/10.1137/0217017. 7

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. DOI: 10.1137/0218012, https://doi.org/10.1137/0218012. 1

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pp. 174–187. IEEE Computer Society Press, 1986. DOI: 10.1109/SFCS.1986.47. 1, 48

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 171–185. Springer, Heidelberg, 1987. DOI: 10.1007/3-540-47721-7_11. 106

[Gol01] O. Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511546891, http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol1.html. 2, 47

[GOP+22] C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model). In *EUROCRYPT 2022*, vol. 13276 of *LNCS*, pp. 397–426. Springer, 2022. DOI: 10.1007/978-3-031-07085-3_14, Full version available at https://eprint.iacr.org/2021/1393.pdf. 15, 16

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 339–358. Springer, Heidelberg, 2006. DOI: 10.1007/11761679_21. 106

[GPP+16] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In *ACM CCS 2016*, pp. 1626–1638. ACM Press, 2016. DOI: 10.1145/2976749.2978353. 25

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pp. 197–206. ACM Press, 2008. DOI: 10.1145/1374376.1374407. 67, 74, 75

[GQ88] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *EUROCRYPT'88*, vol. 330 of *LNCS*, pp. 123–128. Springer, Heidelberg, 1988. DOI: 10.1007/3-540-45961-8_11. 4

[GRBG18] B. Gras, K. Razavi, H. Bos, and C. Giuffrida. Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks. In *USENIX Security 2018*, pp. 955–972. USENIX Association, 2018. 22

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 305–326. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-49896-5_11. 106

[GS86] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *18th ACM STOC*, pp. 59–68. ACM Press, 1986. DOI: 10.1145/12130.12137. 2

[GSM15] D. Gruss, R. Spreitzer, and S. Mangard. Cache template attacks: Automating attacks on inclusive last-level caches. In *USENIX Security 2015*, pp. 897–912. USENIX Association, 2015. 22

[GSW13] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 75–92. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-40041-4_5. 70, 74

[GVW15] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *47th ACM STOC*, pp. 469–477. ACM Press, 2015. DOI: 10.1145/2746539.2746576. 70, 74

[GYCH18] Q. Ge, Y. Yarom, D. Cock, and G. Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018. DOI: 10.1007/s13389-016-0141-6. 22

[HGS01] N. Howgrave-Graham and N. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, 23(3):283–290, 2001. 10, 19, 25

[HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *TCC 2017, Part I*, vol. 10677 of *LNCS*, pp. 341–371. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-70500-2_12. 108, 119

[HJ10] N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In *EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 235–256. Springer, Heidelberg, 2010. DOI: 10.1007/978-3-642-13190-5_12. 27, 30, 72

[HL10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010. DOI: 10.1007/978-3-642-14303-8, https://doi.org/10.1007/978-3-642-14303-8. 2, 47

[HLC+18] Z. Huang, J. Lai, W. Chen, M. H. Au, Z. Peng, and J. Li. Hedged nonce-based public-key encryption: Adaptive security under randomness failures. In *PKC 2018, Part I*, vol. 10769 of *LNCS*, pp. 253–279. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-76578-5_9. 44

[HLR21] J. Holmgren, A. Lombardi, and R. D. Rothblum. Fiat-shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pp. 750–760. ACM, 2021. DOI: 10.1145/3406325.3451116, https://doi.org/10.1145/3406325.3451116. 112, 119

[IAIES14] G. Irazoqui Apecechea, M. S. Inci, T. Eisenbarth, and B. Sunar. Wait a minute! A fast, cross-VM attack on AES. In *RAID*, pp. 299–319, 2014. 22

[IKOS07] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *39th ACM STOC*, pp. 21–30. ACM Press, 2007. DOI: 10.1145/1250790.1250794. 15, 46, 106, 107, 108, 115, 116, 117, 119

[IPSW06] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits II: Keeping secrets in tamperable circuits. In *EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 308–327. Springer, Heidelberg, 2006. DOI: 10.1007/11761679_19. 44

[JSSS20] J. Jancar, V. Sedlacek, P. Svenda, and M. Sys. Minerva: The curse of ECDSA nonces. *IACR TCHES*, 2020(4):281–308, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8684. 10

[JT12] M. Joye and M. Tunstall. *Fault analysis in cryptography*, vol. 147 of *Information Security and Cryptography*. Springer, 2012. 42

[Kat21] S. Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In *CRYPTO 2021, Part II*, vol. 12826 of *LNCS*, pp. 580–610, Virtual Event, 2021. Springer, Heidelberg. DOI: 10.1007/978-3-030-84245-1_20. 8, 112, 119

[KD20] M. Kansal and R. Dutta. Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression. In *AFRICACRYPT 20*, vol. 12174 of *LNCS*, pp. 281–300. Springer, Heidelberg, 2020. DOI: 10.1007/978-3-030-51938-4_14. 74

[KDK+14] Y. Kim, R. Daly, J. Kim, C. Fallin, J. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ISCA*, pp. 361–372. IEEE Computer Society, 2014. 42

[KG20] C. Komlo and I. Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, vol. 12804 of *Lecture Notes in Computer Science*, pp. 34–65. Springer, 2020. DOI: 10.1007/978-3-030-81652-0_2, https://doi.org/10.1007/978-3-030-81652-0_2. 12, 68, 70

[KKM08] A. H. Koblitz, N. Koblitz, and A. Menezes. Elliptic curve cryptography: The serpentine course of a paradigm shift. Cryptology ePrint Archive, Report 2008/390, https://eprint.iacr.org/2008/390. 23

[KKW18] J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pp. 525–537. ACM Press, 2018. DOI: 10.1145/3243734.3243805. 46, 107, 108, 112, 119, 121, 129, 132

[KLP17] E. Kiltz, J. Loss, and J. Pan. Tightly-secure signatures from five-move identification protocols. In *ASIACRYPT 2017, Part III*, vol. 10626 of *LNCS*, pp. 68–94. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-70700-6_3. 6

[KLS18] E. Kiltz, V. Lyubashevsky, and C. Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *EUROCRYPT 2018, Part III*, vol. 10822 of *LNCS*, pp. 552–586. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-78372-7_18. 6, 8, 46, 49, 54, 75, 78, 79

[KM15] N. Koblitz and A. J. Menezes. The random oracle model: a twenty-year retrospective. *Des. Codes Cryptogr.*, 77(2-3):587–610, 2015. DOI: 10.1007/s10623-015-0094-2, https://doi.org/10.1007/s10623-015-0094-2. 8

[KMO90] J. Kilian, S. Micali, and R. Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *CRYPTO'89*, vol. 435 of *LNCS*, pp. 545–546. Springer, Heidelberg, 1990. DOI: 10.1007/0-387-34805-0_47. 48

[KMOS21] 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*, pp. 608–625. IEEE Computer Society Press, 2021. DOI: 10.1109/SP40001.2021.00067. 12

[KMP16] E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO 2016, Part II*, vol. 9815 of *LNCS*, pp. 33–61. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-53008-5_2. 4, 5, 6, 8, 49, 56, 58, 62, 122

[Koc96] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96*, vol. 1109 of *LNCS*, pp. 104–113. Springer, Heidelberg, 1996. DOI: 10.1007/3-540-68697-5_9. 10

[KSV13] D. Karaklajic, J. Schmidt, and I. Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. VLSI Syst.*, 21(12):2295–2306, 2013. DOI: 10.1109/TVLSI.2012.2231707, https://doi.org/10.1109/TVLSI.2012.2231707. 42

[KW03] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM CCS 2003*, pp. 155–164. ACM Press, 2003. DOI: 10.1145/948109.948132. 6

[LCKO19] J. Lee, J. Choi, J. Kim, and H. Oh. SAVER: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Report 2019/1270, https://eprint.iacr.org/2019/1270. 13, 106, 110

[LD99] J. C. López-Hernández and R. Dahab. Fast multiplication on elliptic curves over GF(2^m) without precomputation. In *CHES'99*, vol. 1717 of *LNCS*, pp. 316–327. Springer, Heidelberg, 1999. DOI: 10.1007/3-540-48059-5_27. 28

[LDK+19] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 11, 68, 73, 74

[LFKN90] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pp. 2–10. IEEE Computer Society Press, 1990. DOI: 10.1109/FSCS.1990.89518. 2

[Lin17a] Y. Lindell. Fast secure two-party ECDSA signing. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 613–644. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-63715-0_21. 67, 81

[Lin17b] Y. Lindell. How to simulate it - A tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pp. 277–346. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-57048-8_6, https://doi.org/10.1007/978-3-319-57048-8_6. 2

[LL12] F.-H. Liu and A. Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO 2012*, vol. 7417 of *LNCS*, pp. 517–532. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-32009-5_30. 45

[LN13] M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In *CT-RSA 2013*, vol. 7779 of *LNCS*, pp. 293–309. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-36095-4_19. 20

[LN17] V. Lyubashevsky and G. Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT 2017, Part I*, vol. 10210 of *LNCS*, pp. 293–323. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-56620-7_11. 13, 106, 110, 111

[LN18] Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *ACM CCS 2018*, pp. 1837–1854. ACM Press, 2018. DOI: 10.1145/3243734.3243788. 67

[LNTW19] B. Libert, K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge elementary databases with more expressive queries. In *PKC 2019, Part I*, vol. 11442 of *LNCS*, pp. 255–285. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-17253-4_9. 70, 74

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, vol. 7881 of *LNCS*, pp. 35–54. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-38348-9_3. 103

[LSG+17] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In *USENIX Security 2017*, pp. 557–574. USENIX Association, 2017. 22

[LTT20] Z.-Y. Liu, Y.-F. Tseng, and R. Tso. Cryptanalysis of a round optimal lattice-based multisignature scheme. Cryptology ePrint Archive, Report 2020/1172, https://eprint.iacr.org/2020/1172. 74

[LYG+15] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy*, pp. 605–622. IEEE Computer Society Press, 2015. DOI: 10.1109/SP.2015.43. 22

[Lyu08] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC 2008*, vol. 4939 of *LNCS*, pp. 162–179. Springer, Heidelberg, 2008. DOI: 10.1007/978-3-540-78440-1_10. 6

[Lyu09] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 598–616. Springer, Heidelberg, 2009. DOI: 10.1007/978-3-642-10366-7_35. 6, 13, 67, 71, 74

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 738–755. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-29011-4_43. 13, 67, 71, 74, 78, 79, 85

[Lyu19] V. Lyubashevsky. Lattice-based zero-knowledge and applications. CIS 2019, https://crypto.sjtu.edu.cn/cis2019/slides/Vadim.pdf. 69, 71, 78

[LZ19] Q. Liu and M. Zhandry. Revisiting post-quantum Fiat-Shamir. In *CRYPTO 2019, Part II*, vol. 11693 of *LNCS*, pp. 326–355. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26951-7_12. 8

[MBA+21] R. Merget, M. Brinkmann, N. Aviram, J. Somorovsky, J. Mittmann, and J. Schwenk. Raccoon attack: Finding and exploiting most-significant-bit-oracles in TLS-DH(E). In *USENIX Security 2021*, pp. 213–230. USENIX Association, 2021. 10

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, pp. 2111–2128. ACM Press, 2019. DOI: 10.1145/3319535.3339817. 106

[Mel07] N. Meloni. New point addition formulae for ECC applications. In *WAIFI*, vol. 4547 of *Lecture Notes in Computer Science*, pp. 189–201. Springer, 2007. 40

[Mic00] S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000. DOI: 10.1137/S0097539795284959, https://doi.org/10.1137/S0097539795284959. 106

[MJ19] C. Ma and M. Jiang. Practical lattice-based multisignature schemes for blockchains. *IEEE Access*, 7:179765–179778, 2019. DOI: 10.1109/ACCESS.2019.2958816, https://doi.org/10.1109/ACCESS.2019.2958816. 68, 69, 74

[MNPV99] D. M'Raïhi, D. Naccache, D. Pointcheval, and S. Vaudenay. Computational alternatives to random number generators. In *SAC 1998*, vol. 1556 of *LNCS*, pp. 72–80. Springer, Heidelberg, 1999. DOI: 10.1007/3-540-48892-8_6. 44

[MO09] M. Medwed and E. Oswald. Template attacks on ECDSA. In *WISA 08*, vol. 5379 of *LNCS*, pp. 14–27. Springer, Heidelberg, 2009. 25

[Mon87] P. L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987. 20, 23

[MOR01] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *ACM CCS 2001*, pp. 245–254. ACM Press, 2001. DOI: 10.1145/501983.502017. 68, 86

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 700–718. Springer, Heidelberg, 2012. DOI: 10.1007/978-3-642-29011-4_41. 70, 101, 102, 103, 104

[MP13] D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 21–39. Springer, Heidelberg, 2013. DOI: 10.1007/978-3-642-40041-4_2. 77

[MPSW19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019. DOI: 10.1007/s10623-019-00608-x, https://doi.org/10.1007/s10623-019-00608-x. 12, 68

[MSEH20] D. Moghimi, B. Sunar, T. Eisenbarth, and N. Heninger. TPM-FAIL: TPM meets timing and lattice attacks. In *USENIX Security 2020*, pp. 2057–2073. USENIX Association, 2020. 10, 20

[MSM+16] H. Morita, J. C. N. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata. On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks. In *ICISC 2015*, Lecture Notes in Computer Science, pp. 20–35. Springer, 2016. 45

[MWLD10] C. Ma, J. Weng, Y. Li, and R. H. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptogr.*, 54(2):121–133, 2010. DOI: 10.1007/s10623-009-9313-z, https://doi.org/10.1007/s10623-009-9313-z. 68

[NAB+19] M. Naehrig, E. Alkim, J. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 108, 113, 131

[NKDM03] A. Nicolosi, M. N. Krohn, Y. Dodis, and D. Mazières. Proactive two-party signatures for user authentication. In *NDSS 2003*. The Internet Society, 2003. 68, 69, 72, 85

[NRS21] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In *CRYPTO 2021, Part I*, vol. 12825 of *LNCS*, pp. 189–221, Virtual Event, 2021. Springer, Heidelberg. DOI: 10.1007/978-3-030-84242-0_8. 12, 68, 70

[NRSW20] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In *ACM CCS 2020*, pp. 1717–1731. ACM Press, 2020. DOI: 10.1145/3372297.3417236. 68, 70, 106, 110, 128, 130

[NS02] P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002. DOI: 10.1007/s00145-002-0021-3. 20, 25

[NS03] P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptogr.*, 30(2):201–217, 2003. 25

[NT12] P. Q. Nguyen and M. Tibouchi. Lattice-based fault attacks on signatures. In *Fault Analysis in Cryptography*, Information Security and Cryptography, pp. 201–220. Springer, 2012. DOI: 10.1007/978-3-642-29656-7_12, https://doi.org/10.1007/978-3-642-29656-7_12. 19

[Oka93] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO'92*, vol. 740 of *LNCS*, pp. 31–53. Springer, Heidelberg, 1993. DOI: 10.1007/3-540-48071-4_3. 6

[OLR18] T. Oliveira, J. C. López-Hernández, and F. Rodríguez-Henríquez. The Montgomery ladder on binary elliptic curves. *Journal of Cryptographic Engineering*, 8(3):241–258, 2018. DOI: 10.1007/s13389-017-0163-8. 23

[OO98] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *CRYPTO'98*, vol. 1462 of *LNCS*, pp. 354–369. Springer, Heidelberg, 1998. DOI: 10.1007/BFb0055741. 8, 49

[OST06] D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA 2006*, vol. 3860 of *LNCS*, pp. 1–20. Springer, Heidelberg, 2006. DOI: 10.1007/11605805_1. 22

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, vol. 1592 of *LNCS*, pp. 223–238. Springer, Heidelberg, 1999. DOI: 10.1007/3-540-48910-X_16. 126

[Pas03] R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 316–337. Springer, Heidelberg, 2003. DOI: 10.1007/978-3-540-45146-4_19. 8, 85, 112, 119

[PBY17] P. Pessl, L. G. Bruinderink, and Y. Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In *ACM CCS 2017*, pp. 1843–1855. ACM Press, 2017. DOI: 10.1145/3133956.3134023. 22

[Ped92] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, vol. 576 of *LNCS*, pp. 129–140. Springer, Heidelberg, 1992. DOI: 10.1007/3-540-46766-1_9. 80, 129

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO 2010*, vol. 6223 of *LNCS*, pp. 80–97. Springer, Heidelberg, 2010. DOI: 10.1007/978-3-642-14623-7_5. 75

[Per16] T. Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. Signal, revision 1, https://signal.org/docs/specifications/xeddsa/. 11, 12, 41

[PK15] OASIS Standard: PKCS #11 Cryptographic Token Interface Base Specification Version 2.40, http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf. 105

[PK22] OASIS Standard: PKCS #11 Cryptographic Token Interface Current Mechanisms Specification Version 3.0, https://docs.oasis-open.org/pkcs11/pkcs11-curr/v3.0/csprd01/pkcs11-curr-v3.0-csprd01.pdf. 106

[Poi00] D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC 2000*, vol. 1751 of *LNCS*, pp. 129–146. Springer, Heidelberg, 2000. DOI: 10.1007/978-3-540-46588-1_10. 112

[Por13] T. Pornin. Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). RFC 6979, https://tools.ietf.org/html/rfc6979. 11

[PS00a] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. DOI: 10.1007/s001450010003. 8, 62, 73, 90

[PS00b] G. Poupard and J. Stern. Fair encryption of RSA keys. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 172–189. Springer, Heidelberg, 2000. DOI: 10.1007/3-540-45539-6_13. 106

[PSS+18] D. Poddebniak, J. Somorovsky, S. Schinzel, M. Lochter, and P. Rosler. Attacking Deterministic Signature Schemes using Fault Attacks. In *Euro S&P 2018*, pp. 338–352. IEEE, 2018. 11, 41, 43, 49

[qTE19] The qTESLA Team. *Submission to NIST's post-quantum project (2nd round): lattice-based digital signature scheme qTESLA*, version 2.7, Available at https://qtesla.org/. 11

[RBV17] O. Reparaz, J. Balasch, and I. Verbauwhede. Dude, is my code constant time? In *DATE*, pp. 1697–1702. IEEE, 2017. 40

[RCB16] J. Renes, C. Costello, and L. Batina. Complete addition formulas for prime order elliptic curves. In *EUROCRYPT 2016, Part I*, vol. 9665 of *LNCS*, pp. 403–428. Springer, Heidelberg, 2016. DOI: 10.1007/978-3-662-49890-3_16. 23, 40

[RJH+19] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay, and S. Bhasin. Exploiting determinism in lattice-based signatures: Practical fault attacks on pqm4 implementations of NIST candidates. In *ASIACCS 19*, pp. 427–440. ACM Press, 2019. DOI: 10.1145/3321705.3329821. 11, 41

[RP17] Y. Romailler and S. Pelissier. Practical Fault Attack against the Ed25519 and EdDSA Signature Schemes. In *FDTC 2017*, pp. 17–24, 2017. DOI: 10.1109/FDTC.2017.12. 11, 41, 43, 49

[RS17] J. Renes and B. Smith. qDSA: Small and secure digital signatures with curve-based Diffie-Hellman key pairs. In *ASIACRYPT 2017, Part II*, vol. 10625 of *LNCS*, pp. 273–302. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-70697-9_10. 20

[RY10] T. Ristenpart and S. Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS 2010*. The Internet Society, 2010. 44

[Rya18] K. Ryan. Return of the hidden number problem. *IACR TCHES*, 2019(1):146–168, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7337. 10, 20

[Rya19] K. Ryan. Hardware-backed heist: Extracting ECDSA keys from qualcomm's TrustZone. In *ACM CCS 2019*, pp. 181–194. ACM Press, 2019. DOI: 10.1145/3319535.3354197. 20

[SAB+20] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. 108, 113, 131

[SB18] N. Samwel and L. Batina. Practical fault injection on deterministic signatures: The case of EdDSA. In *AFRICACRYPT 18*, vol. 10831 of *LNCS*, pp. 306–321. Springer, Heidelberg, 2018. DOI: 10.1007/978-3-319-89339-6_17. 11, 41, 43, 49

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, vol. 435 of *LNCS*, pp. 239–252. Springer, Heidelberg, 1990. DOI: 10.1007/0-387-34805-0_22. 4, 8, 19, 68

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. DOI: 10.1007/BF00196725. 4, 10, 46, 128, 130

[Sch16] B. Schmidt. [curves] EdDSA specification. https://moderncrypto.org/mail-archive/curves/2016/000768.html, 2016. 11, 41

[Sha90] A. Shamir. IP=PSPACE. In *31st FOCS*, pp. 11–15. IEEE Computer Society Press, 1990. DOI: 10.1109/FSCS.1990.89519. 2

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pp. 124–134. IEEE Computer Society Press, 1994. DOI: 10.1109/SFCS.1994.365700. 12

[SM16] R. Susella and S. Montrasio. A compact and exception-free ladder for all short weierstrass elliptic curves. In *CARDIS*, vol. 10146 of *Lecture Notes in Computer Science*, pp. 156–173. Springer, 2016. 40

[SS01] D. R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP 01*, vol. 2119 of *LNCS*, pp. 417–434. Springer, Heidelberg, 2001. DOI: 10.1007/3-540-47719-5_33. 68

[SS19] P. Schwabe and D. Sprenkels. The complete cost of cofactor $h = 1$. In *INDOCRYPT 2019*, vol. 11898 of *LNCS*, pp. 375–397. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-35423-7_19. 40

[Sta96] M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT'96*, vol. 1070 of *LNCS*, pp. 190–199. Springer, Heidelberg, 1996. DOI: 10.1007/3-540-68339-9_17. 13, 106

[STV+16] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy*, pp. 526–545. IEEE Computer Society Press, 2016. DOI: 10.1109/SP.2016.38. 68, 71

[TE19] R. Toluee and T. Eghlidos. An efficient and secure ID-based multi-proxy multi-signature scheme based on lattice. Cryptology ePrint Archive, Report 2019/1031, https://eprint.iacr.org/2019/1031. 68, 69, 74

[TheYY] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, https://www.sagemath.org. 34

[TLT19] R. Tso, Z. Liu, and Y. Tseng. Identity-based blind multisignature from lattices. *IEEE Access*, 7:182916–182923, 2019. DOI: 10.1109/ACCESS.2019.2959943, https://doi.org/10.1109/ACCESS.2019.2959943. 68, 69, 74

[TSSK20] W. A. Torres, R. Steinfeld, A. Sakzad, and V. Kuchta. Post-quantum linkable ring signature enabling distributed authorised ring confidential transactions in blockchain. Cryptology ePrint Archive, Report 2020/1121, https://eprint.iacr.org/2020/1121. 74

[TT18] A. Takahashi and M. Tibouchi. New bleichenbacher records: Parallel implementation. https://github.com/security-kouza/new-bleichenbacher-records, 2018. 38

[TT19] A. Takahashi and M. Tibouchi. Degenerate fault attacks on elliptic curve parameters in OpenSSL. In *IEEE EuroS&P 2019*, pp. 371–386. IEEE, 2019. DOI: 10.1109/EuroSP.2019.00035, Full version available at https://eprint.iacr.org/2019/400.pdf. 15

[TTA18a] A. Takahashi, M. Tibouchi, and M. Abe. New Bleichenbacher records: Fault attacks on qDSA signatures. *IACR TCHES*, 2018(3):331–371, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7278. 15, 20, 25, 26, 30, 31, 32, 34, 35, 37, 38, 60

[TTA18b] A. Takahashi, M. Tibouchi, and M. Abe. New Bleichenbacher Records: Fault Attacks on qDSA Signatures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):331–371, 2018. DOI: 10.13154/tches.v2018.i3.331-371, Full version available at https://eprint.iacr.org/2018/396.pdf.

[TTMH02] Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Hiyauchi. Cryptanalysis of block ciphers implemented on computers with cache. In *International Symposium on Information Theory and Its Applications*, 2002. 22

[TuHGB18] N. Tuveri, S. ul Hassan, C. P. García, and B. B. Brumley. Side-channel analysis of SM2: A late-stage featurization case study. In *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, pp. 147–160. ACM, 2018. DOI: 10.1145/3274694.3274725, https://doi.org/10.1145/3274694.3274725. 28

[TZ21] A. Takahashi and G. Zaverucha. Verifiable encryption from MPC-in-the-Head. Cryptology ePrint Archive, Report 2021/1704, https://eprint.iacr.org/2021/1704.pdf. 14, 108, 109, 112, 113, 114, 117, 119, 121, 122, 124, 126, 127, 129, 131

[uHGDL+20] S. ul Hassan, I. Gridin, I. M. Delgado-Lozano, C. P. García, J.-J. Chi-Domínguez, A. C. Aldaya, and B. B. Brumley. Déjà vu: Side-channel analysis of mozilla's NSS. In *ACM CCS 2020*, pp. 1887–1902. ACM Press, 2020. DOI: 10.1145/3372297.3421761. 10

[Unr15] D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, pp. 755–784. Springer, Heidelberg, 2015. DOI: 10.1007/978-3-662-46803-6_25. 8, 112

[Unr17] D. Unruh. Post-quantum security of Fiat-Shamir. In *ASIACRYPT 2017, Part I*, vol. 10624 of *LNCS*, pp. 65–95. Springer, Heidelberg, 2017. DOI: 10.1007/978-3-319-70694-8_3. 8

[vSY15] J. van de Pol, N. P. Smart, and Y. Yarom. Just a little bit more. In *CT-RSA 2015*, vol. 9048 of *LNCS*, pp. 3–21. Springer, Heidelberg, 2015. DOI: 10.1007/978-3-319-16715-2_1. 25

[Wag02] D. Wagner. A generalized birthday problem. In *CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 288–303. Springer, Heidelberg, 2002. DOI: 10.1007/3-540-45708-9_19. 21, 27, 31, 68, 72

[WSBS20] S. Weiser, D. Schrammel, L. Bodner, and R. Spreitzer. Big numbers - big troubles: Systematically analyzing nonce leakage in (EC)DSA implementations. In *USENIX Security 2020*, pp. 1767–1784. USENIX Association, 2020. 20

[Yar16] Y. Yarom. Mastik: A micro-architectural side-channel toolkit. *Retrieved from School of Computer Science Adelaide: http://cs. adelaide. edu. au/yval/Mastik*, 16, 2016. 29

[YAZ+19] R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 147–175. Springer, Heidelberg, 2019. DOI: 10.1007/978-3-030-26948-7_6. 74

[YC22] Yubico YubiHSM2 Guide: Backing Up Key Material, https://developers.yubico.com/YubiHSM2/Usage_Guides/YubiHSM2_for_ADCS_Guide/Backing_Up_Key_Material.html. 105

[YF14] Y. Yarom and K. Falkner. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *USENIX Security 2014*, pp. 719–732. USENIX Association, 2014. 20, 22

[YFT20] M. Yan, C. W. Fletcher, and J. Torrellas. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *USENIX Security 2020*. USENIX Association, 2020. 22

[YJ00] S. Yen and M. Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. Computers*, 49(9):967–970, 2000. DOI: 10.1109/12.869328, https://doi.org/10.1109/12.869328. 45

[YY98] A. Young and M. Yung. Auto-recoverable auto-certifiable cryptosystems. In *EUROCRYPT'98*, vol. 1403 of *LNCS*, pp. 17–31. Springer, Heidelberg, 1998. DOI: 10.1007/BFb0054114. 106

[ZCD+19] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, and V. Kolesnikov. Picnic. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 11, 12, 41, 42, 46, 63

[ZCD+20] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales. Picnic. Technical report, National Institute of Standards and Technology, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. 119, 125