Compact Ring Signatures from Learning With Errors

Rohit Chatterjee [*] Sanjam Garg [†] Mohammad Hajiabadi [‡] Dakshita Khurana [§]
Xiao Liang[¶] Giulio Malavolta [‖] Omkant Pandey [**] Sina Shiehian [††]

May 4, 2022

Abstract

Ring signatures allow a user to sign a message on behalf of a "ring" of signers, while hiding the true identity of the signer. As the degree of anonymity guaranteed by a ring signature is directly proportional to the size of the ring, an important goal in cryptography is to study constructions that minimize the size of the signature as a function of the number of ring members.

In this work, we present the first compact ring signature scheme (i.e., where the size of the signature grows logarithmically with the size of the ring) from the (plain) learning with errors (LWE) problem. The construction is in the standard model and it does not rely on a common random string or on the random oracle heuristic. In contrast with the prior work of Backes *et al.* [EUROCRYPT'2019], our scheme does not rely on bilinear pairings, which allows us to show that the scheme is post-quantum secure assuming the quantum hardness of LWE.

At the heart of our scheme is a new construction of compact and statistically witness indistinguishable ZAP arguments for NP $\cap$ coNP, that we show to be sound based on the plain LWE assumption. Prior to our work, statistical ZAPs (for all of NP) were known to exist only assuming *sub-exponential* LWE. We believe that this scheme might find further applications in the future.

[*] Stony Brook University. Email: rochatterjee@cs.stonybrook.edu
[†] University of California, Berkeley and NTT Research. Email: sanjamg@berkeley.edu
[‡] University of Waterloo. Email: mdhajiabadi@gmail.com
[§] University of Illinois Urbana-Champaign. Email: dakshita@illinois.edu
[¶] Stony Brook University. Email: liang1@cs.stonybrook.edu
[‖] Max Planck Institute for Security and Privacy. Email: giulio.malavolta@hotmail.it
[**] Stony Brook University. Email: omkant@cs.stonybrook.edu
[††] University of California Berkeley and Stony Brook University. Email: shiayan@umich.edu

1 Introduction

In a *ring signature* scheme, (introduced in [RT01]) a user can sign a message with respect to a *ring* of public keys. The ring can be arbitrarily chosen by the signer and the verification keys that populate the ring can be sampled locally by each user, i.e., no central coordination is required. No user or entity should be able to tell which user in the ring actually produced a given signature — a property referred to as *anonymity*. This is complemented with the standard notion of *unforgeability* for signatures, which in this case requires that no user outside a specified ring should be able to produce valid signatures on behalf of this ring. A salient feature is the *online* or *setup-free* generation of ring signatures, which requires that signatures can be generated without any prior interaction between members of the ring. Ring signatures and their variants have found natural applications related to whistleblowing, authenticating leaked information, and more recently to cryptocurrencies [TSS$^+$18, Noe15].

There is a sizeable body of work [LPQ18, PS19a, BLO18, BDH$^+$19] that construct ring signatures under various definitions and hardness assumptions. As the degree of anonymity guaranteed by the ring signature is directly proportional to the size of the ring, an important property of ring signatures becomes *compactness*, which requires that signatures only have a logarithmic (or lower) dependence on the size of the ring. Recently, the work of [BDH$^+$19] provided a compact ring signature construction in the plain model (i.e., not needing a common random string or a setup). Their construction assumes the existence of the following: noninteractive witness indistinguishable proofs or NIWIs (which are known only from bilinear pairing based assumptions or indistinguishability obfuscation), somewhere perfectly binding (SPB) hashes, public key encryption with oblivious public key generation and pseudorandom ciphertexts, and a standard signature scheme. While most of these primitives are known under a variety of cryptographic assumptions including LWE, unfortunately, we currently do not know any constructions of NIWI proofs from LWE (please see the technical overview for related discussion).

This leads to the natural question of whether NIWIs are necessary to construct compact ring signatures in the plain model. The looming threat of quantum computers makes this question particularly pressing, since we would lose our only candidate construction to quantum attacks (due to the reliance on bilinear maps). We therefore ask the following open question:

Can we obtain compact (logarithmic size) ring signatures from the standard (polynomial) hardness of the learning with errors (LWE) problem?

1.1 Our Results

Our main contribution resolves the open problem stated above. In other words, we obtain a ring signature construction from *plain* or *standard* LWE, i.e., only assuming polynomial hardness of the LWE problem with polynomial modulus-to-noise ratio. Our result is obtained as follows:

ZAPs for NP $\cap$ coNP. The first key step to our construction of ring signatures is realization of a new argument system that we call *relaxed ZAPs for extended* $\mathsf{NP} \cap \mathsf{coNP}$. These are akin to ZAPs but with a few additional restrictions, and can also be viewed as a generalization of (non-adaptive) ZAPs for languages in $\mathsf{NP} \cap \mathsf{coNP}$. We show how to construct these ZAPs from the plain (polynomially-hard) LWE. This is in contrast with the known constructions of ZAPs for NP [BFJ$^+$20, GJJM20, LVW19] that assume *subexponential* hardness of LWE. Our ZAP construction also enjoys several other attractive properties such as statistical witness

indistinguishability and proof compactness; which we expect will make them useful in other application contexts. We defer further exposition to our technical overview.

Compact ring signatures from LWE. Next, we show that our notion of relaxed ZAPs, along with SPB hash schemes and a special public key encryption scheme, is sufficient to construct compact ring signatures. All these components have constructions from plain LWE. Thus, we obtain the *first* construction of compact ring signatures in the plain model from purely post-quantum assumptions in the literature. In addition, we investigate security in the *fully quantum* setting, where the adversary can query the signing oracle on a superposition of messages. Towards this goal, we give a construction that retains unforgeability and anonymity in this setting.

1.2 Background

Fiat-Shamir transformation, trapdoor Σ-protocols, and correlation intractability. A trapdoor Σ-protocol [CCH$^+$19] for a language L is a 3-move (honest verifier) zero-knowledge protocol between a prover and a verifier, where the prover tries to convince the verifier about the veracity of a statement x. The protocol is instantiated with an *extractable* commitment scheme where there is an extraction trapdoor td which allows extracting the plaintexts in the commitments. In the first move of the protocol, the prover commits to a string a and sends this commitment $\mathsf{Com}(a)$ to the verifier. Next, in the second move, the verifier sends a challenge b to the prover. In the final round, based on the challenge b, the prover computes the final message and sends it to the verifier. The distinctive property of trapdoor sigma protocols is that, for a false statement x, there is at most one *bad challenge* b^* which lets a malicious prover to successfully complete its proof, and this bad challenge is efficiently computable given a. Consequently, given the extraction trapdoor td, the bad challenge can be efficiently computed from $\mathsf{Com}(a)$.

The Fiat-Shamir transformation [FS86] can convert a trapdoor Σ-protocol (indeed any Σ-protocol) to a noninteractive protocol in the random oracle model. The way it works is that the prover evaluates a hash function on its first message to compute the challenge b, and then proceeds to send the full proof generated using this challenge to the verifier. To argue soundness, notice that since the bad challenge is unique and the hash function is modeled as a random oracle, a malicious prover has a negligible chance of finding a first message such that the hash of it equals the bad challenge b^*.

A line of work [CCH$^+$19, PS19b] builds a special type of hash functions called *correlation intractable* (CI) hash functions, to securely instantiate the Fiat-Shamir heuristic in the plain model for trapdoor Σ-protocols, thus turning them into noninteractive protocols in the CRS model. In particular, the LWE-based CI hash functions in [PS19b] allow building noninteractive zero knowledege protocols for all of NP from the LWE assumption. Informally, a hash function is CI for a class of circuits if for any circuit C in the class, given a random hash key k, it is hard to find an input z whose image under the hash function equals $C(z)$. We can securely replace random oracles with CI hash functions when we apply the Fiat-Shamir transformation to trapdoor Σ-protocols. To see this, notice that since in trapdoor Σ-protocols the bad challenge is efficiently computable given the prover's first message, a malicious prover that can find a first message which gets mapped to the bad challenge is breaking the correlation intractability of the hash function.

Somewhere perfectly binding hash. A somewhere perfectly binding (SPB) hash is similar to a Merkle tree [Mer87]: it can compress a large database of N records into a small digest. In a SPB hash with private local openings, binding holds perfectly for a single hidden index $i \in [N]$. In more detail, the SPB hash key generation algorithm takes as input a binding index $i \in [N]$ and outputs a pair of keys (hk, shk). The hash

key hk can be used to generate a digest. The secret key shk, can be used to generate a short opening for the ith record in the database. Perfect binding says that a valid opening for the ith location of the database uniquely determines the value of that location. Also, the hash key hk is index hiding, i.e, it computationally hides the binding index i.

Compact ring signatures of [BDH$^+$19]. The construction of [BDH$^+$19] is based on four ingredients: a noninteractive witness indistinguishable argument system NIWI, a public key encryption scheme PKE, a standard signature scheme Sig, and a somewhere perfectly binding hash scheme SPB. In this scheme, each verification key consists of two components: a uniformly chosen public key pk (not generated through the key generation algorithm of PKE) and a standard signature verification key vk. The signing key consists of sk, the corresponding signing key for vk. To sign a message m using signing key sk_i corresponding to verification key $\mathsf{VK_i} = (vk_i, pk_i)$, and on behalf of ring $\mathsf{R} = (\mathsf{VK}_1, \cdots, \mathsf{VK}_\ell)$, the signer

- first generates a standard signature σ for message m as $\sigma \leftarrow \mathsf{Sig.Sign}(sk_i, m)$,
- encrypts σ under pk_i with random coins r to get ciphertext $c \leftarrow \mathsf{PKE.Enc}(pk_i, \sigma; r)$,
- generates a binding SPB key pair for index i, $(hk, shk) \leftarrow \mathsf{SPB.Gen}(i)$,
- hashes the ring R with hk to get a short digest $h := \mathsf{SPB.Hash}(hk, \mathsf{R})$,
- creates an opening for the ith location $\tau \leftarrow \mathsf{SPB.Open}(hk, shk, \mathsf{R}, i)$,
- generates a NIWI proof π which using the short opening τ proves existence of a verification key $\mathsf{VK}_i = (vk_i, pk_i)$ in the ring R such that under pk_i, c encrypts a valid signature verifiable with vk_i,
- finally, it publishes (π, c, hk) as the signature for message m.

To prove unforgeability, we switch to a hybrid where we generate each pk_i with a corresponding secret key sk_i. Consequently, perfect binding of SPB and soundness of NIWI imply that given a forgery (π, c, hk) for ring R, there exists a sk_i such that $\mathsf{PKE.Dec}(sk_i, c)$ is valid forgery against Sig. Proving anonymity involves techniques similar to [NY90].

1.3 Technical Overview

ZAP instead of NIWI. As already mentioned, the issue that prevents [BDH$^+$19] from basing their construction solely on LWE is their reliance on NIWIs. Our starting point is the observation of [BKM06] which proposes using two-message public coin argument systems, known in the literature as ZAPs [DN00], instead of NIWIs. Namely, we can add a ZAP first message ρ to each verification key. The signer now picks the lexicographically smallest verification key $\mathsf{VK}_1 = (vk_1, pk_1, \rho_1) \in \mathsf{R}$ in the ring and uses ρ_1 to generate a proof. Using LWE-based ZAPs constructed in [BFJ$^+$20, GJJM20, LVW19], this approach gives us LWE based compact ring signatures. However, there is a major caveat: none of the LWE-based ZAPs mentioned above are based on polynomial hardness assumptions. They all need super-polynomial hardness of LWE (in fact subexponential hardness if the goal is to achieve conventional λ bits of security). Therefore, using lattice based ZAPs for NP seems to be unsatisfactory as the resulting construction would rely on qualitatively stronger super-polynomial hardness assumptions whereas [BDH$^+$19] is based on only polynomial hardness.

ZAPs for $\mathsf{NP} \cap \mathsf{coNP}$. Our next insight is that we may not need ZAPs for all of NP. Assume that in the forgery $\Sigma^* = (\pi^*, c^*, hk^*)$, the ciphertext c^* is guaranteed to be encrypted under one of the public keys

pk_i. In the unforgeability game we can generate pk_is with corresponding secrets sk_i. In this case, given sk_i, checking that Σ^* is a forgery can be done efficiently, i.e., sk_i is a witness for the fact that Σ^* is not a valid signature. Therefore, ZAPs for $\mathsf{NP} \cap \mathsf{coNP}$ might suffice for our application.

It turns out that for $\mathsf{NP} \cap \mathsf{coNP}$ we can build ZAPs based on the polynomial hardness of LWE. We will now describe a ZAP protocol for any arbitrary language $L \in \mathsf{NP} \cap \mathsf{coNP}$. The ZAP that we describe here is constructed by following the general framework of converting a Σ-protocol to a noninteractive protocol using a CI hash function [CCH$^+$19, PS19b]. More specifically, we describe a two round commitment scheme and use it to instantiate [CCH$^+$19, PS19b]. Our commitment scheme is defined with respect to the complement language $\overline{L}$. To commit to a bit b and generate the second commitment message, the sender specifies a statement x. The receiver can recover the committed bit, if, when generating the first message it specified a (non-)witness $\overline{w}$ for the fact that $x \in \overline{L}$. If $x \notin \overline{L}$, the committed bit is hidden. The commitment scheme works as follows:

- The receiver sends an arbitrary bitstring $\overline{w}$ bit-by-bit via the first message of a statistically sender private (1 out of 2) OT, $\mathsf{OT1}(\overline{w})$.
- The sender garbles the following circuit: on input $\overline{w}$, if $\overline{w}$ is a witness for $x \in \overline{L}$, output b, otherwise, output 0. The sender sends the garbled circuit along with an OT second message containing the labels $\mathsf{OT2}(\{\mathsf{lbl}_{i,0}, \mathsf{lbl}_{i,1}\})$.

We instantiate [CCH$^+$19, PS19b] for language L with our two round *witness extractable commitment* for $\overline{L}$, to get a ZAP for $\mathsf{NP} \cap \mathsf{coNP}$: the verifier sends the commitment first message along with a key for a CI hash function, the prover uses the hash key and the commitment scheme to proceed as in [CCH$^+$19]. Recall that to apply the transformation of [CCH$^+$19, PS19b], we have to make sure that when $x \notin L$, the commitments used in the underlying Σ-protocol are extractable given the extraction trapdoor. If $x \notin L$, the verifier can (undetectably) switch to generating the first commitment message using a non-witness $\overline{w_x}$ for $x \in \overline{L}$ and this will let the soundness proof go through.

Unfortunately, when trying to use our $\mathsf{NP} \cap \mathsf{coNP}$ ZAPs to build ring signatures we encounter two issues:

1. The ZAPs that we need for our ring signatures need to be adaptively sound. However, the ZAPs that we just constructed are only selectively sound. This is because, in the soundness reduction we have to switch to a commitment first message that depends on a non-witness $\overline{w}$ for the statement x (depends on a $\overline{w}$ such that $(x, \overline{w}) \in \overline{R}$).
2. We assumed the ciphertext c^* in the forgery is a valid encryption under one of the public keys of the ring. This may not be true. In particular, $\overline{L}$ might not be in NP.

The first issue seems relatively easy to resolve. Our ZAP construction already achieves a weak notion of adaptivity that is sufficient for our purpose: as long as the non-witness $\overline{w}$ is fixed in advance, the cheating prover cannot come up with a valid proof for a statement x where $(x, \overline{w}) \in \overline{R}$. In our case, the non-witnesses which are the secret keys corresponding to pk_is are clearly fixed in advance.

Extending the complement language. For the second issue, our solution is to *extend* $\overline{L}$ to a language $\widetilde{L} \in \mathsf{NP}$ and use a witness extractable commitment for this extended language $\widetilde{L}$. In more detail, given a forgery Σ^* for a ring R^* of size ℓ, we define statement $x = (\Sigma^*, \mathsf{R}^*)$. For a witness $\widetilde{w} = (sk_1, \cdots, sk_\ell)$ we say that $(x, \widetilde{w}) \in \widetilde{L}$, if each sk_i is a valid secret key for pk_i, and decrypting c^* with any sk_j does not yield a valid standard signature corresponding to vk_j. Accordingly, in the unforgeability game, we can generate each public key pk_i with a corresponding secret sk_i, and put these secret keys inside the witness extractable

commitment first message. With this approach we encounter a new obstacle: the size of the commitment first message-and consequently the size of verifier's first message in the ZAP scheme-scales at least linearly with the maximum number of members in a ring. However, the number of members in a ring can be arbitrarily large.

One secret key, multiple public keys. To overcome this problem we use a public key cryptosystem which can generate multiple public keys having a single secret key. Using such a public key scheme, the first commitment message now only needs to hold one short secret key. Luckily, public key cryptosystems with this property already exist in the literature. In particular, Regev's cryptosystem [Reg05] already has the ability to generate multiple pseudorandom public keys having a single uniformly chosen secret key. This cryptosystem also has another appealing feature: it has sparse valid public keys. In other words, a randomly chosen public key does not have a corresponding secret key (except with negligible probability). In the ring signature context, this sparseness property will be helpful in proving the anonymity of the scheme. Specifically, given a signature Σ for a ring R, if at least one verification key in R is generated honestly, or in particular if at least one public key $pk_i \in \mathsf{R}$ is chosen uniformly at random, then $(\Sigma, \mathsf{R}) \notin \widetilde{L}$, and therefore, the commitment scheme is hiding.

Compactness through malicious circuit private FHE. The ZAP that we have constructed so far seems to satisfy the soundness and witness indistinguishibility properties that we need in the application to our ring signature scheme. However, upon a closer look, it seems that our resulting ZAP scheme suffers from a major flaw. Namely, the size of our ring signatures is linear in the size of the ring. This is because in the witness extractable commitment scheme, each commitment contains a garbling of a circuit that computes on each verification key in the ring separately. Specifically, while the size of the circuit for checking membership in L is independent of the size of the ring (thanks to the properties of SPB hashing), circuits for checking membership in $\widetilde{L}$, and even the relevant statements, have size that depend on the ring. It seems that to overcome this, we additionally need some guarantee of *compactness* in our witness extractable commitment. Our final idea is to use a fully homomorphic encryption scheme [Gen09] to build such a compact witness extractable commitment. The construction is as follows:

- The receiver generates an FHE key pair $(\mathsf{FHE}.pk, \mathsf{FHE}.sk)$ and sends a ciphertext $ct \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE}.pk, \widetilde{w})$ encrypting an arbitrary string $\widetilde{w}$.
- The sender homomorphically evaluates the following circuit on ct: on input $\widetilde{w}$, if $\widetilde{w}$ is a witness for $x \in \widetilde{L}$, output b, otherwise, output 0. The sender sends the evalauted ciphertext ct_{eval}.

Observe now that the compactness of the commitment scheme follows from the compactness of the FHE scheme. Clearly, if the receiver encrypts a non-witness for x, it can recover the committed bit b using the secret key $\mathsf{FHE}.sk$. For this commitment scheme to be hiding, it is sufficient that in the FHE scheme, the evaluated ciphertext hides the circuit that has been evaluated on it, even if the initial FHE ciphertext and public key are maliciously generated. Fortunately, FHE schemes satisfying the aforementioned *malicious circuit privacy* have already been constructed from LWE [OPP14, BD18].

1.4 Related Existing Work

The initial construction of ring signatures by Rivest, Shamir and Kalai [RT01] is in the random oracle model. Several subsequent constructions [AOS02, BGLS03, HS03] were also given in the ROM. Constructions in the standard model were first obtained concurrently by Chow, Liu, Wei and Yuen [CWLY06] and Bender,

Katz and Morselli [BKM06]. Several works also construct schemes in the CRS model [SW07, BH18, SS10]. Brakerski and Kalai [BK10] gave a construction based on the SIS problem in the standard model, and there are subsequent lattice-based constructions [BLO18, TSS+18] that give more practical constructions (these works actually construct *linkable* ring signatures). Park and Sealfon [PS19a] give certain constructions based on SIS and others based on verifiable random functions that satisfy new and interesting definitions of repudiability and claimability that they develop. All of these constructions give ring signatures linear in the ring size.

Dodis et al [DKNS04] gave the first sublinear size ring signatures in the ROM. Since then, constructing logarithmic size ring signatures in the ROM been the focus of many works [GK15, LLNW16, LPQ18, EZS+19, BKP20, LNS21]. [KKW18] construct a compact ring signature using a Merkle tree, based on their NIZKPoK argument in the ROM. In the CRS model, [CGS07, Gha13, Gon17, LNPY20] build sublinear ring signatures under various assumptions. In the plain model, Backes et al [BHKS18] construct sublinear ring signatures using a primitive called signatures with *flexible* public key, and Malavolta and Schroder [MS17] construct constant size ring signatures under a knowledge of exponent assumption, which is unfalsifiable. Finally, as mentioned, Backes et al [BDH+19] construct the first logarithmic size ring signatures in the plain model under standard and falsifiable assumptions, namely DDH or LWE along with NIWI proofs.

2 Preliminaries

We denote the security parameter by λ. For any $\ell \in \mathbb{N}$, we denote the set of the first ℓ positive integers by $[\ell]$. For a set S, $x \leftarrow S$ denotes sampling a uniformly random element x from S.

2.1 Learning With Errors

We recall the Learning With Errors (LWE) problem, and its hardness based on worst-case lattice problems.

For a positive integer dimension n and modulus q, and an error distribution χ over $\mathbb{Z}$, the LWE distribution and decision problem are defined as follows. For an $\mathbf{s} \in \mathbb{Z}^n$, the LWE distribution $A_{\mathbf{s},\chi}$ is sampled by choosing a uniformly random $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an error term $e \leftarrow \chi$, and outputting $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e) \in \mathbb{Z}_q^{n+1}$.

Definition 2.1. The decision-LWE$_{n,q,\chi}$ problem is to distinguish, with non-negligible advantage, between any desired (but polynomially bounded) number of independent samples drawn from $A_{\mathbf{s},\chi}$ for a single $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and the same number of *uniformly random* and independent samples over $\mathbb{Z}_q^{n+1}$.

A standard instantiation of LWE is to let χ be a *discrete Gaussian* distribution over $\mathbb{Z}$ with parameter $r = 2\sqrt{n}$. A sample drawn from this distribution has magnitude bounded by, say, $r\sqrt{n} = \Theta(n)$ except with probability at most 2^{-n}, and hence this tail of the distribution can be entirely removed. For this parameterization, it is known that LWE is at least as hard as *quantumly* approximating certain "short vector" problems on n-dimensional lattices, in the worst case, to within $\tilde{O}(q\sqrt{n})$ factors [Reg05, PRS17]. Classical reductions are also known for different parameterizations [Pei09, BLP+13].

Fix a dimension $n = \mathrm{poly}(\lambda)$. For the rest of this paper, when we refer to hardness of LWE, we mean hardness of LWE with polynomial modulus-to-noise ratio against polynomial sized adversaries, i.e., polynomial hardness of LWE$_{n,q,\chi}$ where, q is a polynomial in n, and χ is the error distribution described in the previous paragraph.

2.2 Correlation Intractable Hash Functions

We borrow the following the definition of CI hash functions from [PS19b] verbatim.

Definition 2.2. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a family of circuits, i.e., a set of circuits for each λ. A hash function family $\mathsf{Hash} = (\mathsf{Gen}, \mathsf{Eval})$ is *correlation intractable (CI) for* $\mathcal{C}$ if for every non-uniform polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}$ there exists a negligible function $\nu(\lambda)$ such that for every $C \in \mathcal{C}_\lambda$,

$$\Pr_{\substack{k \leftarrow \mathsf{Hash.Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}_\lambda(k)}} [\mathsf{Hash.Eval}(k, x) = C(x)] \leq \nu(\lambda) \ .$$

We will also use CI hash construction of [PS19b].

Theorem 2.3 ([PS19b]). *Assuming hardness of LWE, there exists a polynomial $m = m(\lambda)$ such that, for every family of polynomial sized circuits $\mathcal{C}$ with output size at least m bits, there exists a hash function family which is CI for $\mathcal{C}$. Furthermore, the key generation algorithm in this hash function family simply outputs a uniformly random key from its key space.*

2.3 Public Key Encryption

Similar to [BDH+19], we need a public key encryption scheme which has pseudorandom ciphertexts and public keys. For our application, we also require additional properties.

Definition 2.4. A public key encryption scheme is a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{GenWithKey}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Valid})$, with the following interfaces, where for each security parameter $\lambda \in \mathbb{N}$, PK_λ, SK_λ and CT_λ are three sets where the uniform distribution is efficiently sampleable,

- $\mathsf{GenWithKey}(sk)$, on input a secret key $sk \in \mathsf{SK}_\lambda$ outputs a public key $pk \in \mathsf{PK}_\lambda$.
- $\mathsf{Enc}(pk, b)$, on input a public key pk, and a message $b \in \{0, 1\}$, outputs a ciphertext $ct \in \mathsf{CT}_\lambda$.
- $\mathsf{Dec}(sk, ct)$, on input a secret key sk and a ciphertext ct, outputs a bit b.
- $\mathsf{Valid}(pk, sk)$, on input a public pk and a secret key sk, either accepts or rejects.

We consider the following properties:

1. *Completeness*: for any $\lambda \in \mathbb{N}$, any key pair (pk, sk) such that $\mathsf{Valid}(pk, sk)$ accepts, and any message b,
$$\Pr[\mathsf{Dec}(sk, ct) = b] = 1,$$
where $ct \leftarrow \mathsf{Enc}(pk, m)$. Furthermore, for any $\lambda \in \mathbb{N}$,
$$\Pr_{\substack{sk \leftarrow \mathsf{SK}_\lambda \\ pk \leftarrow \mathsf{GenWithKey}(sk)}} [\mathsf{Valid}(pk, sk) \text{ accepts}] = 1.$$
2. *Injectivity of key generation*: for any sk,
$$\Pr_{pk \leftarrow \mathsf{GenWithKey}(sk)} [\exists sk' \neq sk : \mathsf{Valid}(pk, sk') \text{ accepts}] = \mathrm{negl}(\lambda).$$
3. *Pseudorandomness of public keys*: for every $Q = \mathrm{poly}(\lambda)$, the following two distributions
   - first, samples a uniformly random secret key $sk \leftarrow \mathsf{SK}_\lambda$, then outputs $\{pk_i \leftarrow \mathsf{GenWithKey}(sk)\}_{i \in [Q]}$
   - outputs L uniformly random public keys $\{pk_i \leftarrow \mathsf{PK}_\lambda\}_{i \in [Q]}$,

are computationally indistinguishable.

4. *Almost uniform ciphertexts*: for every message b, the output of the following two distributions
   - first, samples a uniformly random public key $pk \leftarrow \mathsf{PK}_\lambda$, then, outputs $(pk, \mathsf{Enc}(pk, b))$,
   - first, samples a uniformly random public key $pk \leftarrow \mathsf{PK}_\lambda$, then, chooses a uniformly random ciphertext $ct \leftarrow \mathsf{CT}_\lambda$ and outputs (pk, ct),

   are statistically indistinguishable.

Consider Regev's public key cryptosystem [Reg05]. For some appropriate parameters $n = \mathrm{poly}(\lambda), q = \mathrm{poly}(n), m \geq 2n \log q$, $B \ll q/4$, a secret key in this scheme is a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and valid public keys for secret $\mathbf{s}$ are generated as

$$pk := \mathbf{A} = \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{s}^t \bar{\mathbf{A}} + \mathbf{e}^t \end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m},$$

where, $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{e}$ is chosen from some B-bounded distribution. For this cryptosystem we define the following validity check algorithm

- $\mathsf{Valid}(pk = \mathbf{A}, sk = \mathbf{s})$: Accept iff $|(\mathbf{s}^t, -1) \cdot \mathbf{A}|_\infty \leq B$.

Theorem 2.5. *Assuming hardness of LWE, there exists a public key encryption scheme satisfying all the properties in Definition 2.4.*

Proof. We briefly argue that equipped with algorithm Valid, Regev's cryptosystem satisfies all the properties in Definition 2.4. When q is a prime number, injectivity of key generation is an implication of Lemma 5.3 in [GPV08]. The rest of the properties are already established in [Reg05]. □

2.4 Blum's Raw Protocol

Here, we formally define and state the properties of the raw version of Blum's sigma protocol [Blu87]. In this abstraction, the prover does not use any commitment or encryption scheme to hide its first message and therefore, the protocol does not satisfy a conventional zero knowledge property. Using a commitment scheme, this protocol can be converted into an honest verifier zero knowledge protocol.

Definition 2.6. Let $L \in \mathsf{NP}$ be a language with a corresponding relation R. Blum's raw protocol for L, is a tuple of PPT algorithms
$\Pi = (\mathsf{P1}, \mathsf{P2}, \mathsf{V}, \mathsf{BadChallenge}, \mathsf{Sim})$ with the following interfaces

- $\mathsf{P1}(x, w)$, on input a statement witness pair $(x, w) \in L$, it outputs a first message a, two substrings (a_0, a_1) of a corresponding to two subsets of indices S_0, S_1, and an internal state ζ. We implicitly assume (a_0, a_1) uniquely determine the subsets S_0, S_1.
- $\mathsf{P2}(a, b, \zeta)$, on input first message a, a *challenge* bit $b \in \{0, 1\}$, and internal state ζ, it outputs a second message c.
- $\mathsf{V}(x, b, a_b, c)$, on input an instance x, a challenge bit $b \in \{0, 1\}$, an *opening* a_b, and a response c, it either accepts or rejects.
- $\mathsf{BadChallenge}(a)$, on input a first message a, it outputs a bad-challenge bit $b \in \{0, 1\}$.

- $\mathsf{Sim}(b, x)$, takes as input a challenge bit b, and an instance x, and outputs two strings a_b and c.

These algorithms satisfy the following properties:

1. *Completeness*: for any $(x, w) \in L$, any $\ell \in \mathbb{N}$, and any $b \in \{0, 1\}$,

$$\Pr[\mathsf{V}(x, b, a_b, c) \text{ accepts }] = 1,$$

   where, $(a, (a_0, a_1), \zeta) \leftarrow \mathsf{P1}(x, w)$, and $c \leftarrow \mathsf{P2}(a, b, \zeta)$.

2. *Soundness*: if $x \notin L$, then, for any two subsets of indices S_0, S_1, bit $b = 1 - \mathsf{BadChallenge}(a)$, and any c,

$$\Pr[\mathsf{V}(x, b, a_b, c) \text{ rejects }] = 1,$$

   where, a_b denotes the subset of a specified by indices in S_b.

3. *Zero knowledge*: for any $b \in \{0, 1\}$ and any $(x, w) \in R$ the following two distributions,

   - outputs (b, a_b, c), where $(a, (a_0, a_1), \zeta) \leftarrow \mathsf{P1}(x, w)$, and $c \leftarrow \mathsf{P2}(a, b, \zeta)$
   - outputs $(b, \mathsf{Sim}(b, x))$

   are identical.

Blum's raw protocol exists unconditionally for any language $L \in \mathsf{NP}$ [Blu87].

2.5 Maliciously Circuit Private FHE

We review the definition of maliciously circuit private FHE.

Definition 2.7 ([OPP14]). A maliciously circuit private leveled FHE scheme is a tuple of algorithms $\mathsf{FHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Sim})$, where, except for Sim the rest of the algorithms are PPT, having the following interfaces

- $\mathsf{Gen}(1^\lambda, 1^d)$, given a security parameter $\lambda \in \mathbb{N}$ and a depth parameter $d \in \mathbb{N}$, outputs a public and private key pair (pk, sk).
- $\mathsf{Enc}(pk, b)$, given a public key pk and a message $b \in \{0, 1\}$, outputs a ciphertext $ct \in \{0, 1\}^{\ell_{ct}(\lambda, d)}$.
- $\mathsf{Eval}((ct_1, \cdots, ct_k), C; r)$, given k ciphertexts $ct_1, \cdots, ct_k$, a boolean circuit $C : \{0, 1\}^k \to \{0, 1\}$, and random coins r, outputs an evaluated ciphertext $ct_{eval} \in \{0, 1\}^{\ell_{eval}}$.
- $\mathsf{Dec}(sk, ct)$, given a secret key sk and a ciphertext ct, outputs a bit $b \in \{0, 1\}$.
- $\mathsf{Sim}(pk^*, (ct_1^*, \cdots, ct_k^*), b)$, on input a (not necessarily honestly generated) public-key pk^* and k (not necessarily honestly generated) ciphertexts $ct_1^* \in \{0, 1\}^{\ell_{ct}(\lambda, d)}, ..., ct_k^* \in \{0, 1\}^{\ell_{ct}(\lambda, d)}$, and a bit b, outputs a simulated ciphertext ct_{sim}.

We consider FHE schemes that satisfy the following properties:

1. *Completeness*: for every $\lambda, d \in \mathbb{N}$, every circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d and every input $m \in \{0,1\}^k$,
$$\Pr[\mathsf{Dec}(sk, ct_{eval}) = C(m)] = 1,$$
where, $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^d)$, $ct_i \leftarrow \mathsf{Enc}(pk, m_i)$ for every $i \in [\ell]$, and $ct_{eval} \leftarrow \mathsf{Eval}((ct_1, \cdots, ct_\ell), C)$.

2. *Compactness*: there exists fixed polynomials $\ell_{eval} = \ell_{eval}(\lambda, d)$ and $\ell_{rand} = \ell_{rand}(\lambda, d)$, such that evaluated ciphertexts have size $\ell_{eval}(\lambda, d)$ and the size of the randomness used in Eval algorithm is $\ell_{rand}(\lambda, d)$, i.e., the size of evaluated ciphertexts and the size of randomness in the evaluation algorithm only depend on the depth of the circuit being evaluated.

3. *Pseudorandomness of public keys*: the public key pk output by the Gen algorithm is pseudorandom.

4. *Pseudorandomness of ciphertexts*: for every non-uniform polynomial-size adversary $\mathcal{A}$, every $d \in \mathbb{N}$ and every $b \in \{0,1\}$, the probabilities
$$\Pr[\mathcal{A}(pk, ct) = 1],$$
in the following two experiments differ by only $\mathrm{negl}(\lambda)$:
   - in experiment 1, $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^d)$, $ct \leftarrow \mathsf{Enc}(pk, b)$
   - in experiment 2, $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^d)$, $ct \leftarrow \{0,1\}^{\ell_{ct}(\lambda,d)}$

5. *Malicious circuit privacy*: for every (not necessarily honestly generated) public-key pk^*, every $k \in \mathbb{N}$, and every k (not necessarily honestly generated) ciphertexts $ct_1^* \in \{0,1\}^{\ell_{ct}(\lambda,d)}, ..., ct_k^* \in \{0,1\}^{\ell_{ct}(\lambda,d)}$, there exists a $m^* \in \{0,1\}^k$ such that, for every circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d,
$$\mathsf{Eval}((ct_1^*, \cdots, ct_k^*), C) \overset{s}{\approx} \mathsf{Sim}(pk^*, (ct_1^*, \cdots, ct_k^*), C(m^*))$$
.

Theorem 2.8 ([OPP14, BD18]). *Assuming hardness of LWE, there exists a maliciously circuit private leveled FHE, where the size of evaluated ciphertexts and secret keys only depend on the security parameter λ.*

2.6 Somewhere Perfectly Binding Hash

We next define the notion of somewhere perfectly binding hash or SPB schemes, which are very similar to somewhere statistically binding hash schemes (as can be surmised, the only change from the latter is that here we expect the somewhere binding property to hold with probability 1). As in the scheme of [BDH$^+$19], we will only define and employ a slightly weaker primitive denoted as *somewhere perfectly binding hash with private local openings*, which is what we will need for our signature scheme as well. We direct the reader to [BDH$^+$19] for further details. The definition is essentially identical to that in [BDH$^+$19], and is as follows.

Definition 2.9 (SPB Hash). A somewhere perfectly binding hash scheme with private local openings, denoted by SPB, consists of a tuple of probabilistic polynomial time algorithms $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$, with the following syntax:

- $\mathsf{Gen}(1^\lambda, n, \mathsf{ind})$, given a security parameter λ, a database size n, and index ind as input, outputs a hash and secret key pair (hk, shk).
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{db})$, given a hash key hk and database db as input, outputs a hash value h.
- $\mathsf{Open}(hk, shk, \mathsf{db}, \mathsf{ind})$, given a hash key hk, secret key shk, database db and index ind as input, outputs a witness τ.
- $\mathsf{Verify}(hk, h, \mathsf{ind}, x, \tau)$, given as input a hash key hk, a hash value h, an index ind, a value x and a witness τ, either accepts or rejects.

To maintain clarity, we will not explicitly specify the block size of databases as input to Gen, but assume that this is clear from the specific usage and hardwired into the algorithm. We ask that the SPB scheme satisfies the following properties:

1. *Correctness:* for all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, all databases db of size n, and all indices $\mathsf{ind} \in [n]$, we have,

$$\Pr[\mathsf{Verify}(hk, h, \mathsf{ind}, \mathsf{db}_{\mathsf{ind}}, \tau) \text{ accepts }] = 1,$$

   where, $(hk, shk) \leftarrow \mathsf{Gen}(1^\lambda, n, \mathsf{ind})$, $h \leftarrow \mathsf{Hash}(hk, \mathsf{db})$ and $\tau \leftarrow \mathsf{Open}(hk, shk, \mathsf{db}, \mathsf{ind})$.

2. *Efficiency:* any hash keys hk and witnesses τ corresponding to size n databases, are of size $\log(n) \cdot \mathrm{poly}(\lambda)$. Further, for size n databases, Verify can be computed by a circuit of size $\log(n) \cdot \mathrm{poly}(\lambda)$.

3. *Somewhere perfectly binding:* for all $\lambda, n \in \mathbb{N}$, all databases db of size n, all indices $\mathsf{ind} \in [n]$, all purported hashing keys hk, all purported witnesses τ, all h in the support of $\mathsf{Hash}(hk, \mathsf{db})$, if $\mathsf{Verify}(hk, h, \mathsf{ind}, x, \tau)$ accepts, then $x = \mathsf{db}_{\mathsf{ind}}$.

4. *Index hiding,* for any $n \in \mathbb{N}$ and any $\mathsf{ind}_0, \mathsf{ind}_1 \in [n]$,

$$\{hk : (hk, shk) \leftarrow \mathsf{Gen}(1^\lambda, n, \mathsf{ind}_0)\} \overset{c}{\approx} \{hk : (hk, shk) \leftarrow \mathsf{Gen}(1^\lambda, n, \mathsf{ind}_1)\}$$

Theorem 2.10 ([BDH+19]). *Assuming hardness of LWE, there exists a SPB hash.*

2.7 Signature Schemes

Definition 2.11 (Signature Schemes). Let $\mathcal{M} = \{\mathcal{M}_\lambda\}$ be a family of sets where for each λ, $\mathcal{M}_\lambda$ consists of bit-strings of some fixed polynomial size. A signature scheme Sig for $\mathcal{M}$ is described by a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that have the following syntax:

- $\mathsf{Gen}(1^\lambda)$, given a security parameter λ as input, outputs a verification and signing key pair (vk, sk).
- $\mathsf{Sign}(sk, m)$, given a signing key sk and a message $m \in \mathcal{M}_\lambda$ as input, outputs a signature σ.
- $\mathsf{Verify}(vk, m, \sigma)$, given a verification key vk, message $m \in \mathcal{M}_\lambda$ and signature σ as input, either accepts or rejects.

Further, we require that the signature scheme satisfies the following requirements:

1. *Correctness:* for all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}_\lambda$, we have

$$\Pr[\mathsf{Verify}(vk, m, \sigma) \text{ accepts }] = 1,$$

where $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(sk, m)$.

2. *Existential unforgeability:* any PPT adversary $\mathcal{A}$ has negligible advantage in the following experiment:

Experiment $\mathsf{EUF{-}CMA}(\mathcal{A})$**:** This experiment is run by a challenger that proceeds as follows:

- The challenger generates a key pair $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and stores it. It provides vk to $\mathcal{A}$.
- $\mathcal{A}$ can now make signing queries of the form (sign, m). Upon receiving such a query, the challenger computes $\sigma \leftarrow \mathsf{Sign}(sk, m)$ and sends this to $\mathcal{A}$. It also keeps a list of all such queries made by $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs a forgery attempt, namely a purported signature σ^* with respect to a message m^*. The challenger checks if:
  - $\mathcal{A}$ never made a signing query with respect to m^*, and
  - $\mathsf{Verify}(vk, m^*, \sigma^*)$ accepts

  If so, the challenger outputs 1, otherwise, it outputs 0.

The *advantage* of the adversary $\mathcal{A}$ is defined to be $\mathbf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr[\mathsf{EUF{-}CMA}(\mathcal{A}) = 1]$.

Signature schemes based on hardness of LWE are well known in the lattice cryptography literature [CHKP10].

2.8 Ring Signatures

We review the definition of compact ring signatures as presented in [BDH$^+$19].

Definition 2.12 (Compact Ring Signature [BDH$^+$19]). A compact ring signature scheme RS is described by a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that have the following interface:

- $\mathsf{Gen}(1^\lambda, N)$, given a security parameter λ and the maximum number of members in a ring N, outputs a verification and signing key pair $(\mathsf{VK}, \mathsf{SK})$.
- $\mathsf{Sign}(\mathsf{SK}, m, \mathsf{R})$, given a secret key SK, a message $m \in \mathcal{M}_\lambda$. and a list of verification keys (interpreted as a ring) $\mathsf{R} = (\mathsf{VK}_1, \cdots, \mathsf{VK}_\ell)$ as input, outputs a ring signature Σ.
- $\mathsf{Verify}(\mathsf{R}, m, \Sigma)$, given a ring $\mathsf{R} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_\ell)$, message $m \in \mathcal{M}_\lambda$ and ring signature Σ as input, either accepts or rejects.

Further, we require that the ring signature satisfies the following properties:

1. *Completeness:* for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $\ell \leq N$, $i \in [\ell]$ and $m \in \mathcal{M}_\lambda$, we have:

$$\Pr[\mathsf{RS.Verify}(\mathsf{R}, m, \Sigma) \text{ accepts}] = 1,$$

where, $(\mathsf{VK_i}, \mathsf{SK_i}) \leftarrow \mathsf{Gen}(1^\lambda, N)$, $\Sigma \leftarrow \mathsf{Sign}(\mathsf{SK_i}, m, \mathsf{R})$ (where $\mathsf{R} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_\ell)$).

2. *Unforgeability:* for any $N \in \mathbb{N}$, and any $Q = \mathrm{poly}(\lambda)$, any PPT adversary $\mathcal{A}$ has negligible advantage in the following game:

   Experiment $\mathsf{RS{-}Forge}^{\mathsf{Q}}(\mathcal{A})$**:** This experiment is run by a challenger that proceeds as follows:

   - For each $i \in [Q]$, the challenger generates key pairs $(\mathsf{VK_i}, \mathsf{SK_i}) \leftarrow \mathsf{Gen}(1^\lambda, N; r_i)$, and stores these key pairs along with their corresponding randomness. It then sets $\mathcal{VK} = \{\mathsf{VK_1}, \ldots, \mathsf{VK_Q}\}$ and initializes $\mathcal{C} = \emptyset$.
   - The challenger sends $\mathcal{VK}$ to $\mathcal{A}$.
   - $\mathcal{A}$ can now make the following two kinds of queries: signing queries sign to get signatures signed by a particular entity on a particular message with respect to a ring of its choice, and corruption queries $\mathsf{corrupt}$ to corrupt a particular entity. The challenger responds as follows:
     - Signing query $(\mathsf{sign}, i, m, \mathsf{R})$: The challenger first checks if $\mathsf{VK}_i \in \mathsf{R}$. If so, it computes $\Sigma \leftarrow \mathsf{Sign}(\mathsf{SK_i}, m, \mathsf{R})$ and sends this to $\mathcal{A}$. It also keeps a list of all such queries made by $\mathcal{A}$.
     - Corruption query $(\mathsf{corrupt}, i)$: The challenger adds VK_i to the set $\mathcal{C}$ and returns the randomness r_i to $\mathcal{A}$.
   - Finally, $\mathcal{A}$ outputs a forgery attempt, namely a purported ring signature Σ^* with respect to a ring R^* and message m^*. The challenger checks if:
     - $\mathsf{R}^* \subseteq \mathcal{VK} \setminus \mathcal{C}$,
     - $\mathcal{A}$ never made a signing query with respect to m^* and R^* (together, i.e. of the form $(\mathsf{sign}, \cdot, m^*, \mathsf{R}^*)$), and
     - $\mathsf{Verify}(\mathsf{R}^*, m^*, \Sigma^*)$ accepts.

     If so, the challenger outputs 1, otherwise, it outputs 0.

   The *advantage* of the adversary $\mathcal{A}$ is defined to be $\mathbf{Adv}_{\mathsf{RS-Forge}^Q}(\mathcal{A}) = \Pr[\mathsf{RS{-}Forge}^Q(\mathcal{A}) = 1]$.

3. *Anonymity:* for all $Q = \mathrm{poly}(\lambda)$, any PPT adversary $\mathcal{A}$ has negligible advantage in the following game:

   Experiment $\mathsf{RS{-}Anon}^{Q}(\mathcal{A})$**:** This experiment is run by a challenger that proceeds as follows:

   - For each $i \in [Q]$, the challenger generates key pairs $(\mathsf{VK_i}, \mathsf{SK_i}) \leftarrow \mathsf{Gen}(1^\lambda; r_i)$. It sends these key pairs along with their corresponding randomness to $\mathcal{A}$.
   - Eventually $\mathcal{A}$ sends a challenge to the challenger of the form $(\mathsf{R}, m, i_0, i_1)$. We stress that R might have keys that are not generated by the challenger in the previous step. In particular, it might contain maliciously generated keys. The challenger checks if $\mathsf{VK}_{i_0}, \mathsf{VK}_{i_1} \in \mathsf{R}$. If so, it first samples a uniform bit $b \leftarrow \{0, 1\}$, then computes $\Sigma \leftarrow \mathsf{Sign}(\mathsf{SK}_{\mathsf{i_b}}, m, \mathsf{R})$, and sends this to $\mathcal{A}$.
   - $\mathcal{A}$ sends back its guess bit b'. The challenger outputs 1 if $b' = b$, otherwise it outputs 0.

   The $\mathsf{advantage}$ of the adversary $\mathcal{A}$ in this game is defined as

   $$\mathbf{Adv}_{\mathsf{RS-Anon}^Q}(\mathcal{A}) = |\Pr[\mathsf{RS{-}Anon}^Q(\mathcal{A}) = 1] - \frac{1}{2}|.$$

4. *Compactness:* the size of a signature is upper-bounded by a polynomial in λ and $\log N$.

We mention that the unforgeability and anonymity properties defined in Definition 2.12 correspond respectively to the notions of *unforgeability with insider corruption* and *anonymity with respect to full key exposure* presented in [BKM06].

3 Compact Witness Extractable Commitments

In this section we define witness extractable commitments. A witness extractable commitment for a language $L \in \mathsf{NP}$ with corresponding NP relation R is a two round commitment protocol. The receiver's message is generated using a witness w, and the sender's commitment to a bit b is generated using a statement x. Informally speaking, the bit b can be efficiently extracted when $(x, w) \in R$, however, when $x \notin L$, b is statistically hidden.

For our application in this paper, we are interested in witness extractable commitments that are compact. This means that the size of a commitment second message does not depend on the size of the NP verifier circuit (except for maybe its depth).

3.1 Definition

Definition 3.1. Fix a language $L \in \mathsf{NP}$. By R and $\{C_{n,\ell} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}\}_{n,\ell \in N}$ denote the NP relation and the NP verification circuit corresponding to L respectively. Also, let $d = d(n, \ell)$ be the depth of $C_{n,\ell}$. When it is clear from the context, we use d instead of $d(n, \ell)$. A *witness extractable commitment scheme* for L, is a tuple of PPT algorithms $(\mathsf{Com1}, \mathsf{Com2}, \mathsf{Verify}, \mathsf{Extract})$ having the following interfaces:

- $\mathsf{Com1}(1^\lambda, 1^D, w)$, given a security parameter λ, circuit depth upper bound D, and a witness $w \in \{0,1\}^\ell$, outputs the commitment first message $\mathsf{com1} \in \{0,1\}^{\ell_{com1} = \ell_{com1}(\lambda, D, \ell)}$ and a string $\mathsf{st} \in \{0,1\}^{\ell_{st}}$ representing the internal state.
- $\mathsf{Com2}(\mathsf{com1}, x, b; r)$, given a commitment first message $\mathsf{com1}$, a statement x, a bit b to commit, and randomness $r \in \{0,1\}^{\ell_r}$, outputs a commitment $\mathsf{com2} \in \{0,1\}^{\ell_{com2}}$.
- $\mathsf{Verify}(\mathsf{com1}, \mathsf{com2}, x, b, r)$, given a commitment transcript $\mathsf{com1}, \mathsf{com2}$, a statement x, a bit b, and random coins r, it either accepts or rejects.
- $\mathsf{Extract}(\mathsf{com2}, \mathsf{st})$, given a commitment $\mathsf{com2}$ and internal state st, outputs a bit b.

We consider the following properties:

1. *Completeness*: for every $\lambda \in \mathbb{N}$, bit b, every statement x, every witness w, every $D \geq d$,

$$\Pr[\mathsf{Verify}(\mathsf{com1}, \mathsf{com2}, x, b, r) \text{ accepts }] = 1,$$

   where, $\mathsf{com1} \leftarrow \mathsf{Com1}(1^\lambda, 1^D, w)$, $r \leftarrow \{0,1\}^{\ell_r}$, and $\mathsf{com2} \leftarrow \mathsf{Com2}(\mathsf{com1}, x, b; r)$.

2. *Statistical hiding*: if $x \notin L$, then, for any $\ell \in \mathbb{N}$, any $D \geq d(|x|, \ell)$, and any sequence of strings $\{\mathsf{com1}_\lambda \in \{0,1\}^{\ell_{com1}(\lambda, D, \ell)}\}_{\lambda \in \mathbb{N}}$,

$$\mathsf{com1}_\lambda, \mathsf{Com2}(\mathsf{com1}_\lambda, x, 0) \overset{s}{\approx} \mathsf{com1}_\lambda, \mathsf{Com2}(\mathsf{com1}_\lambda, x, 1)$$

3. *Pseudorandomness of first message*: for any w and any D,

$$\mathsf{Com1}(1^\lambda, 1^D, w) \overset{c}{\approx} U_{\ell_{com1}}$$

4. *Extractability*: if $(x, w) \in R$, then, for any bit b, any $D \geq d$, any $\mathsf{com1}, \mathsf{st}$ in the support of $\mathsf{Com1}(1^\lambda, 1^D, w)$, any randomness r, and any $\mathsf{com2}$ such that $\mathsf{Verify}(\mathsf{com1}, \mathsf{com2}, x, b, r)$ accepts,

$$\Pr[\mathsf{Extract}(\mathsf{com2}, \mathsf{st}) = b] = 1.$$

5. *Compactness*: the parameters ℓ_{com2},ℓ_r and ℓ_{st} are upper-bounded by some language-independent fixed polynomials in λ and D. In particular, they are independent of the size of the NP verifier circuit $C_{n,\ell}$ and the size of the statement x.

3.2 Construction

We will use the maliciously circuit private FHE scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval})$ of [OPP14].

Construction 3.2. Here we use the same notation as in Definition 3.1.

- $\mathsf{Com1}(1^\lambda, 1^D, w)$: run $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.Gen}(1^\lambda, 1^{D+1})$. Output $\mathsf{com1} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.pk}, w)$. Keep $\mathsf{st} := \mathsf{FHE.sk}$ as internal state. Notice that for any circuit C of depth D, the circuit constructed in Figure 1 has depth $D + 1$.
- $\mathsf{Com2}(\mathsf{com1}, x, b)$: on input first message com1 and bit b, output

  $$\mathsf{com2} \leftarrow \mathsf{FHE.Eval}(\mathsf{com1}, G_{x,b}; r),$$

  where, $G_{x,b}$ is the circuit defined below and r represents the random coins used in the FHE evaluation algorithm.
- $\mathsf{Extract}(\mathsf{com2}, \mathsf{st} = \mathsf{FHE.sk})$: output $\mathsf{FHE.Dec}(\mathsf{FHE.sk}, \mathsf{com2})$.
- $\mathsf{Verify}(\mathsf{com1}, \mathsf{com2}, x, b, r)$: accept iff com2 is equal to $\mathsf{FHE.Eval}(\mathsf{com1}, G_{x,b}; r)$.

procedure $G_{x,b}(w)$
  if $C(x, w) = 1$ **then**
    Output b
  else
    Output 0

Figure 1: Description of $G_{x,b}$

Extractability and completeness of Construction 3.2 follow immediately from completeness of FHE. Compactness also follows from compactness of FHE. In fact, using the FHE in [OPP14], both ℓ_{st} and ℓ_{com2} only depend on λ. Finally, pseudorandomness of FHE ciphertext and public keys imply pseudorandomness of the first message in Construction 3.2.

Theorem 3.3. *If* FHE *is maliciously circuit private, then, the commitment scheme in Construction 3.2 is statistically hiding.*

Proof. Let $x \notin L$ and let com1 be an arbitrary string. We prove this theorem via a series of hybrids.

Hybrid H_0: This hybrid corresponds to committing to bit 0, i.e., it outputs

$$(\mathsf{com1}, \mathsf{FHE.Eval}(\mathsf{com1}, G_{x,0})).$$

Hybrid H_1: This hybrid outputs

$$(\mathsf{com1}, \mathsf{FHE.Sim}(\mathsf{FHE}.pk, \mathsf{com1}, 0)).$$

Hybrid H_2: This hybrid corresponds to committing bit 1, i.e., it outputs

$$(\mathsf{com1}, \mathsf{FHE.Eval}(\mathsf{com1}, G_{x,1})).$$

Lemma 3.4. *If* FHE *is statistically maliciously circuit private,* $H_0 \overset{s}{\approx} H_1$.

Proof. Since $x \notin L$, $C(x, w) = 0$ for any w. The lemma follows by the statistical malicious circuit privacy of FHE. □

Lemma 3.5. *If* FHE *is statistically maliciously circuit private,* $H_1 \overset{s}{\approx} H_2$.

Proof. Since $x \notin L$, $C(x, w) = 0$ for any w. The lemma follows by the statistical malicious circuit privacy of FHE. □

This completes the proof. □

4 Compact Relaxed ZAPs for Extended NP ∩ coNP

In this section, we define and construct a 2-round public-coin argument system. Our argument system can be viewed as a generalization of ZAPs for NP ∩ coNP. To describe this generalization, first we introduce the notion of *super-complement* of a language. A super-complement of a language is an extension of a subset of the complement of that language. Notably, the complement of a language is a super-complement of it.

Definition 4.1 (Super-Complement). Let L, $\widetilde{L}$ be two languages where the elements of $\widetilde{L}$ are represented as pairs of bit strings. We say $\widetilde{L}$ is a *super-complement* of L, if

$$\widetilde{L} \subseteq (\{0,1\}^* \setminus L) \times \{0,1\}^*,$$

i.e., $\widetilde{L}$ is a super complement of L if for any $x = (x_1, x_2)$,

$$x \in \widetilde{L} \Rightarrow x_1 \notin L.$$

Our argument system is defined for two NP languages $L, \widetilde{L}$, where, $\widetilde{L}$ is a super-complement of L. Notice that, while the complement of L might not be in NP, however we require that $\widetilde{L} \in \mathsf{NP}$. The language $\widetilde{L}$ is used to define the soundness property. Namely, producing a proof for a statement $x = (x_1, x_2) \in \widetilde{L}$, should be hard. We also use the fact that $\widetilde{L} \in \mathsf{NP}$ to mildly strengthen the soundness property. In more detail, instead of having selective soundness where the statement $x \in \widetilde{L}$ is fixed in advance, now, we fix a non-witness $\widetilde{w}$ and let the statement x be adaptively chosen by the malicious prover from all statements which have $\widetilde{w}$ as a witness to their membership in $\widetilde{L}$.

For our application to compact ring signatures, we further require the size of the proofs to be compact with respect to $\widetilde{L}$. Roughly speaking, this means that size of a proof for a statement $x = (x_1, x_2)$ only depends on the size of x_1.

4.1 Definition

Definition 4.2. Let $L, \widetilde{L} \in \mathsf{NP}$ be two languages such that $\widetilde{L}$ is a super complement of L. By R and $\widetilde{R}$ denote the NP relations corresponding to L and $\widetilde{L}$ respectively. Let $\{C_{n,\ell}\}_{n,\ell \in N}$ and $\{\widetilde{C}_{n,\ell}\}_{n,\ell \in N}$ be the NP verification circuits for L and $\widetilde{L}$ respectively. Let $\widetilde{d} = \widetilde{d}(n,\ell)$ be the depth of $\widetilde{C}_{n,\ell}$. A *compact relaxed ZAP* for $L, \widetilde{L}$ is a tuple of PPT algorithms $(\mathsf{V}, \mathsf{P}, \mathsf{Verify})$ having the following interfaces (where $1^n, 1^\lambda$ are implicit inputs to P, Verify):

- $\mathsf{V}(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}})$, given a security parameter λ, statement length n for L, witness length $\widetilde{\ell}$ for $\widetilde{L}$, and NP verifier circuit depth upper-bound $\widetilde{D}$ for $\widetilde{L}$, outputs a first message ρ.
- $\mathsf{P}(\rho, x = (x_1, x_2), w)$, given a string ρ, a statement $(x_1 \in \{0,1\}^n, x_2)$, and a witness w such that $(x_1, w) \in R$, outputs a proof π.
- $\mathsf{Verify}(\rho, x = (x_1, x_2), \pi)$, given a string ρ, a statement x, and a proof π, either accepts or rejects.

We consider the following properties:

1. *Completeness*: for every $(x_1, w) \in L$, every $x_2 \in \{0,1\}^*$, every $\widetilde{\ell} \in \mathbb{N}$, every $\widetilde{D} \geq \widetilde{d}(|x_1| + |x_2|, \widetilde{\ell})$, and every $\lambda \in \mathbb{N}$,
$$\Pr[\mathsf{Verify}(\rho, x = (x_1, x_2), \pi) \text{ accepts }] = 1,$$
where, $\rho \leftarrow \mathsf{V}(1^\lambda, 1^{|x_1|}, 1^{\widetilde{\ell}}, 1^{\widetilde{D}})$ and $\pi \leftarrow \mathsf{P}(\rho, x, w)$.

2. *Public coin*: $\mathsf{V}(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}})$ simply outputs a uniformly random string.

3. *Selective non-witness and adaptive statement soundness*: for every non-uniform polynomial-size "cheating" prover $P^* = \{P^*_\lambda\}$ there exists a negligible function $\nu(\lambda)$ such that for any $n, \widetilde{D} \in \mathbb{N}$ and any non-witness $\widetilde{w} \in \{0,1\}^*$,
$$\Pr_{\substack{\rho \leftarrow \mathsf{V}(1^\lambda, 1^n, 1^{|\widetilde{w}|}, 1^{\widetilde{D}}) \\ (x = (x_1, x_2), \pi^*) \leftarrow P^*_\lambda(\rho)}} [\mathsf{Verify}(\rho, x, \pi^*) \text{ accepts} \wedge \widetilde{D} \geq \widetilde{d}(|x|, |\widetilde{w}|) \wedge (x, \widetilde{w}) \in \widetilde{R}] \leq \nu(\lambda).$$

4. *Statistical witness indistinguishability*: for every (possibly unbounded) "cheating" verifier $V^* = (V^*_1, V^*_2)$, and every $n, \widetilde{\ell}, \widetilde{D} \in \mathbb{N}$ the probabilities
$$\Pr[V^*_2(\rho, x, \pi, \zeta) = 1 \wedge (x, w) \in \mathcal{R} \wedge (x, w') \in \mathcal{R}]$$
in the following two experiments differ only by $\mathrm{negl}(\lambda)$:
   - in experiment 1, $(\rho, x, w, w', \zeta) \leftarrow V^*_1(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}}), \pi \leftarrow \mathsf{P}(\rho, x, w)$
   - in experiment 2, $(\rho, x, w, w', \zeta) \leftarrow V^*_1(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}}), \pi \leftarrow \mathsf{P}(\rho, x, w')$

5. *Compactness*: bit-size of proof π is a fixed polynomial in $n, \widetilde{\ell}, \widetilde{D}, |C|$ and λ. In particular, it is independent of the size of $\widetilde{C}$ and x_2.

4.2 Construction

For languages $L, \widetilde{L}$, we give the tuple of algorithms $(\mathsf{V}, \mathsf{P}, \mathsf{Verify})$ that make up our relaxed ZAP scheme. In the construction we will use the following ingredients:

- A witness extractable commitment $\mathsf{Com} = (\mathsf{Com1}, \mathsf{Com2}, \mathsf{Verify}, \mathsf{Extract})$ for $\widetilde{L}$. We denote the sizes of the first commitment message, second commitment message, the internal state output by $\mathsf{Com1}$, and the randomness for $\mathsf{Com2}$, by ℓ_{com1} ,ℓ_{com2}, ℓ_{st} and ℓ_r respectively.
- Blum's raw protocol $\Pi = (\mathsf{P1}, \mathsf{P2}, V, \mathsf{BadChallenge}, \mathsf{Sim})$ for L. We denote the size of the first and second prover messages by ℓ_{p1} and ℓ_{p2} respectively. For any $\ell \in N$, Π^ℓ means repeating the protocol Π, ℓ times in parallel and interpreting the inputs to the algorithms accordingly. When it is clear from the context, we drop the superscript ℓ.
- A hash family $\mathsf{Hash}^{n,\ell_{rep}} = (\mathsf{Gen}, \mathsf{Eval})$ that for any $n \in \mathbb{N}$, and any polynomial $\ell_{rep} = \ell_{rep}(\lambda)$ that is larger than the polynomial $m(\lambda)$ in Theorem 2.3, is CI for a circuit family $\mathcal{C}^{n,\ell_{rep}}$ which we will define next. The circuit family is defined as

  $$\mathcal{C}^{n,\ell_{rep}} = \{\mathcal{C}_\lambda^{n,\ell_{rep}}\}_{\lambda \in \mathbb{N}},$$

  where for each $\lambda \in \mathbb{N}$,

  $$\mathcal{C}_\lambda^{n,\ell_{rep}} = \{f_{st} : \{0,1\}^{\ell_{rep} \cdot \ell_{p1} \cdot \ell_{com2}} \to \{0,1\}^{\ell_{rep}}\}_{st \in \{0,1\}^{\ell_{st}(\lambda)}},$$

  where, f_{st} is defined as

  $$f_{st}(x) = \Pi^{\ell_{rep}}.\mathsf{BadChallenge}(\mathsf{Com.Extract}(x, st)),$$

  i.e., f_{st} extracts a message from the input commitment and outputs the *bad challenge* corresponding to that message. We will drop the indices n, ℓ_{rep} when they are clear from the context.

Construction 4.3. Let $\ell_{rep} = \ell_{rep}(\lambda)$ be a polynomial that is larger than the polynomial $m(\lambda)$ in Theorem 2.3.

- $\mathsf{V}(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}})$: first, pick a uniformly random commitment first message $\mathsf{com1} \leftarrow \{0,1\}^{\ell_{com1}(\lambda, \widetilde{D}, \widetilde{\ell})}$, then, generate CI-hash key $k \leftarrow \mathsf{Hash}^{n,\ell_{rep}}.\mathsf{Gen}(1^\lambda)$. Output $\rho := (\mathsf{com1}, k)$.
- $\mathsf{P}(\rho = (\mathsf{com1}, k), x = (x_1, x_2), w)$: first, compute

  $$(\mathbf{a}, \{a_{i,b}\}_{i \in [\ell_{rep}], b \in \{0,1\}}, \zeta) \leftarrow \Pi^{\ell_{rep}}.\mathsf{P1}(x_1, w),$$

  and then send

  $$\pi = (\mathsf{com2} = \mathsf{Com.Com2}(\mathsf{com1}, x, \mathbf{a}; \mathbf{r}), I = \mathsf{Hash.Eval}(k, \mathsf{com2}), \mathbf{c} = \Pi.\mathsf{P2}(\mathbf{a}, I, \zeta), \mathbf{r}_I, \mathbf{a}_I)$$

  to the verifier, where, $\mathbf{a}_I = \{a_{i,I_i}\}_{i \in [\ell_{rep}]}$ and $\mathbf{r}_I$ denotes the subset of randomness used to commit to $\mathbf{a}_I$. Also, $\mathsf{com2}_I$ denotes the chunks of $\mathsf{com2}$ which commit to $\mathbf{a}_I$.
- $\mathsf{Verify}(\rho, x = (x_1, x_2), \pi)$: parse $\pi = (\mathsf{com2}, I, \mathbf{c}, \mathbf{r}_I, \mathbf{a}_I)$. Accept iff both $\Pi.\mathsf{V}(x, I, \mathbf{a}_I, \mathbf{c})$ and $\mathsf{Com.Verify}(\mathsf{com1}, \mathsf{com2}_I, x, \mathbf{a}_I, \mathbf{r}_I)$ accept.

Completeness of Construction 4.3 follows directly from the completeness of Π and Com. It is also public coin because the CI hash keys are uniform. Compactness also follows from the compactness of Com, namely, it follows from the fact that ℓ_{com2}, ℓ_{st} and ℓ_r may only polynomially depend on the depth $\widetilde{D}$ of $\widetilde{C}$ and not its size.

Theorem 4.4. *The protocol described in Construction 4.3 satisfies selective non-witness adaptive statement soundness.*

Proof. Fix a non-witness $\widetilde{w} \in \{0,1\}^{\widetilde{\ell}}$ and let $\mathcal{A}$ be an adversary against the selective non-witness adaptive statement soundness of Construction 4.3. Consider the following hybrids:

Hybrid H_0: This is the protocol described in Construction 4.3.

Hybrid H_1: Here, the verifier generates $\mathsf{com1}$ using $\mathsf{Com.Com1}$ instead of sampling it uniformly at random. Namely, it computes $(\mathsf{com1}, st) \leftarrow \mathsf{Com.Com1}(1^\lambda, \widetilde{w})$ and stores st.

Lemma 4.5. *Assuming Construction 3.2 has pseudorandom first messages, $H_0 \overset{c}{\approx} H_1$.*

Proof. This follows directly from the definition of pseudorandomness of first message property in Definition 2.4. □

Lemma 4.6. *Assuming Hash is CI for $\mathcal{C}$, Com is extractable, and Π is sound, the probability that in H_1 the verifier accepts is negligible.*

Proof. Using $\mathcal{A}$ we build an adversary $\mathcal{A}'$ against the CI property of Hash, having at least as much advantage as $\mathcal{A}$. Before receiving a hash key k for Hash, $\mathcal{A}'$ generates $(\mathsf{com1}, st) \leftarrow \mathsf{Com.Com1}(1^\lambda, \widetilde{w})$. The goal of $\mathcal{A}'$ is to find a correlation for Hash with respect to the function $f_{st} \in \mathcal{C}_\lambda$. When $\mathcal{A}$ receives the hash key k, it sends $\rho = (\mathsf{com1}, k)$ to $\mathcal{A}$. Next, $\mathcal{A}$ responds with a pair $(x, \pi = (\mathsf{com2}, I, \mathbf{c}, \mathbf{r}_I, \mathbf{a}_I))$. Finally, $\mathcal{A}'$ outputs $z := \mathsf{com2}$.

Clearly, the distribution of ρ sent by $\mathcal{A}'$ is identical to its distribution in H_1. Now, notice that, if $(x, \widetilde{w}) \in \widetilde{R}$, then, $x_1 \notin L$. Let $\mathbf{a}' := \mathsf{Com.Extract}(\mathsf{com2}, st)$. If $\mathsf{Verify}(\rho, x, \pi)$ accepts, then,

$$(i)\mathsf{Com.Verify}(\mathsf{com1}, \mathsf{com2}, x_1, \mathbf{a}_I, \mathbf{r}_I) \text{ accepts}$$

and,

$$(ii)\Pi.\mathsf{V}(x_1, I, \mathbf{a}_I, \mathbf{c}_I) \text{ accepts.}$$

By extractability of Com, (i) implies that, $\mathbf{a}_I = \mathbf{a}'_I$, where $\mathbf{a}'_I$ denotes the substring of $\mathbf{a}'$ whose indices corresponds to the indices determined by $\mathbf{a}_I$. Consequently, since Π is sound, (ii) implies that, $I = \Pi.\mathsf{BadChallenge}(\mathbf{a}')$. Therefore,

$$\Pr[\mathsf{Verify}(\rho, x, \pi) \text{ accepts }] \leq \Pr[\mathsf{Hash.Eval}(k, \mathsf{com2}) = f_{st}(\mathsf{com2})].$$

This completes the proof of this lemma. □

Observe that, checking $(x, \widetilde{w}) \in \widetilde{R}$ can be done efficiently. Therefore, the theorem follows by Lemma 4.5 and Lemma 4.6. □

We now show that Construction 4.3 is statistically witness indistinguishable. Our proof strategy resembles the one used in [GJJM20].

Theorem 4.7. *The protocol described in Construction 4.3 is statistically witness indsitinguishable.*

Proof. We proceed via a series of hybrids. For each $0 \leq i \leq \ell_{rep}$ and each $0 \leq j \leq 5$ define hybrids $H_{i,j}$ as follows:

Hybrid $H_i = H_{i,0}$: In this hybrid, the first i repetitions of the proof use witness w_2 and the rest use w_1. More concretely, here, for each $j \in [i]$ the prover computes

$$(a_j, (a_{j,0}, a_{j,1}), \zeta_j) \leftarrow \Pi.\mathsf{P1}(x_1, w_2),$$

and for each $j \in [i+1, \ell_{rep}]$ computes,

$$(a_j, (a_{j,0}, a_{j,1}), \zeta_j) \leftarrow \Pi.\mathsf{P1}(x_1, w_1).$$

The rest of H_i proceeds exactly as in P.

Hybrid $H_{i,1}$: Here, the prover first picks a uniformly random bit $b' \leftarrow \{0,1\}$ and then proceeds exactly as in $H_{i,0}$ to compute I. If $b' \neq I_{i+1}$, it restarts, otherwise it continues and finishes the proof as in $H_{i,0}$.

Hybrid $H_{i,2}$: The difference between this hybrid and the previous hybrid is how the prover generates the commitment $\mathsf{com2}_{i+1}$. In this hybrid, the prover commits to $a_{i+1,b'}$ as before. But, instead of committing to the bits of a_{i+1} which are not in $a_{i+1,b'}$, i.e. $a_{i+1} \setminus a_{i+1,b'}$, the prover commits to $\mathbf{0}$.

Hybrid $H_{i,3}$: This hybrid does not use $\Pi.\mathsf{P1}$ to generate a_{i+1}. Instead, it computes $(a_{i+1,b'}, c_{i+1})$ as

$$(a_{i+1,b'}, c_{i+1}) \leftarrow \Pi.\mathsf{Sim}(b', x_1),$$

and implicitly sets the rest of bits of a_{i+1}, i.e., the bits of a_{i+1} not determined by $a_{i+1,b'}$, to zero.

Hybrid $H_{i,4}$: The only difference between this hybrid and the previous one is that, here, the prover computes $a_{i+1}, a_{i+1,0}, a_{i+1,1}, c_{i+1}$ as follows

$$\begin{aligned}(a_{i+1}, (a_{i+1,0}, a_{i+1,1}), \zeta_{i+1}) &\leftarrow \Pi.\mathsf{P1}(x_1, w_2)\\ c_{i+1} &\leftarrow \Pi.\mathsf{P2}(a_{i+1}, b', \zeta_{i+1}).\end{aligned}$$

Hybrid $H_{i,5}$: This hybrid is almost identical to $H_{i,4}$ except that, here, the prover computes $\mathsf{com2}_{i+1}$ as

$$\mathsf{com2}_{i+1} \leftarrow \mathsf{Com.Com2}(\mathsf{com1}, x, a_{i+1}).$$

Notice that $H_{0,0}$ corresponds to generating a proof using w_1 and $H_{\ell_{rep},0}$ corresponds to using w_2 as the witness.

Lemma 4.8. *For every $0 \leq i < \ell_{rep}$, $H_{i,0}$ and $H_{i,1}$ are distributed identically.*

Proof. This follows from the fact that in $H_{i,1}$, b' is chosen uniformly at random. □

Lemma 4.9. *For every $0 \leq i < \ell_{rep}$, $H_{i,1} \overset{s}{\approx} H_{i,2}$.*

Proof. Since $x_1 \in L$, $x \notin \widetilde{L}$. Therefore, the lemma follows by the statistical hiding property of Com. □

Lemma 4.10. *For every $0 \leq i < \ell_{rep}$, $H_{i,2}$ and $H_{i,3}$ are identically distributed.*

Proof. This is a direct implication of the zero knowledge property of Π. □

Lemma 4.11. *For every* $0 \leq i < \ell_{rep}$, $H_{i,3}$ *and* $H_{i,4}$ *are identically distributed.*

Proof. This is a direct implication of the zero knowledge property of Π. □

Lemma 4.12. *For every* $0 \leq i < \ell_{rep}$, $H_{i,4} \overset{s}{\approx} H_{i,5}$.

Proof. Since $x_1 \in L$, $x \notin \widetilde{L}$. Therefore, the lemma follows by the statistical hiding property of Com. □

Lemma 4.13. *For every* $0 \leq i < \ell_{rep}$, $H_{i,5}$ *and* $H_{i+1,0}$ *are identically distributed.*

Proof. This follows from the fact that b' is chosen uniformly at random. □

This completes the proof. □

5 Compact LWE-based Ring Signature Scheme

In this section, we present our compact ring signature scheme. First, we briefly list the ingredients in our construction:

- A standard signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with $\mathsf{EUF}-\mathsf{CMA}$ security.
- A public key encryption scheme $\mathsf{PKE} = (\mathsf{GenWithKey}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Valid})$ as defined in Definition 2.4.
- A somewhere perfectly binding hash function $\mathsf{SPB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ with private local openings.
- A compact relaxed ZAP scheme $\mathsf{ZAP} = (\mathsf{V}, \mathsf{P}, \mathsf{Verify})$ as described in section 4.

Next, we define the languages L and $\widetilde{L}$ that we will instantiate our relaxed ZAP construction for. The language L is identical to the language $\mathcal{L}$ used in the ring signature construction of [BDH$^+$19]. For a statement $y_1 = (m, c, hk, h)$ and witness $w = (\mathsf{VK} = (vk, pk, \rho), i, \tau, \sigma, r_c)$, define relations R_1, R_2 and R_3 as follows:

$$(y_1, w) \in R_1 \Leftrightarrow \mathsf{SPB.Verify}(hk, h, i, \mathsf{VK}, \tau) \text{ accepts}$$
$$(y_1, w) \in R_2 \Leftrightarrow \mathsf{PKE.Enc}(pk, (\sigma, vk); r_c) = c$$
$$(y_1, w) \in R_3 \Leftrightarrow \mathsf{Sig.Verify}(vk, m, \sigma) \text{ accepts}$$

Next, define the relation R' as

$$R' := R_1 \cap R_2 \cap R_3.$$

Let L' be the language corresponding to R'. For statements of the form $(m, c_1, c_2, hk_1, hk_2, h_1, h_2)$, define the language L as

$$L = \{(m, c_1, c_2, hk_1, hk_2, h_1, h_2) | (m, c_1, hk_1, h_1) \in L' \vee (m, c_2, hk_2, h_2) \in L'\}.$$

Now, we define the language $\widetilde{L}$ and prove that it is a super-complement of L. Let $x_2 = \mathsf{R} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_\ell)$, $y = (y_1, x_2)$, and $\widetilde{w} = s$. Define the following relations:

$$(y, \widetilde{w}) \in R_4 \Leftrightarrow \forall j \in [\ell] : \mathsf{PKE.Valid}(pk_j, s) \text{ accepts} \wedge h = \mathsf{SPB.Hash}(hk, \mathsf{R})$$
$$(y, \widetilde{w}) \in R_5 \Leftrightarrow \mathsf{PKE.Dec}(s, c) = (\sigma, vk) \wedge \mathsf{Sig.Verify}(vk, m, \sigma) \text{ accepts}$$
$$\wedge\, \exists \mathsf{VK} \in \mathsf{R} : \mathsf{VK} = (vk, pk, \rho) \text{ for some } pk \text{ and } \rho$$

where, for each $j \in [\ell]$, $\mathsf{VK}_j = (vk_j, pk_j, \rho_j)$. Let L_4, L_5 be the languages corresponding to R_4, R_5 respectively.

Define further the relation $\widehat{R}$ according to

$$\widehat{R} := R_4 \setminus R_5,$$

and let $\widehat{L}$ be the corresponding language. Finally, for statements of the form
$x = (x_1 = (m, c_1, c_2, hk_1, hk_2, h_1, h_2), x_2 = \mathsf{R})$, let $\widetilde{L}$ be the language given by

$$\widetilde{L} = \{(m, c_1, c_2, hk_1, hk_2, h_1, h_2, \mathsf{R}) | (m, c_1, hk_1, h_1, \mathsf{R}) \in \widehat{L} \wedge (m, c_2, hk_2, h_2, \mathsf{R}) \in \widehat{L}\}.$$

Given the properties of the SPB and PKE we can quickly prove the following lemma.

Lemma 5.1. *If* SPB *is somewhere perfectly binding and* PKE *is complete,* $\widetilde{L}$ *is a super-complement of* L*.*

Proof. Consider the statement

$$x = (x_1 = (m, c_1, c_2, hk_1, hk_2, h_1, h_2), x_2 = \mathsf{R}),$$

and let

$$y_1 = (m, c_1, hk_1, h_1) \text{ and}$$
$$y_1' = (m, c_2, hk_2, h_2).$$

We show that for every $w = (\mathsf{VK} = (vk, pk, \rho), i, \tau, \sigma, r_c)$ and every $\widetilde{w} = s$,

$$(y_1, w) \in R' \wedge ((y_1, x_2), \widetilde{w}) \in R_4 \Rightarrow ((y_1, x_2), \widetilde{w}) \in R_5.$$

If $(y_1, w) \in R_1$ and $((y_1, x_2), \widetilde{w}) \in R_4$, then, by the somewhere perfectly binding of SPB, $\mathsf{VK} \in \mathsf{R}$. Now, notice that by the completeness of PKE, since $\mathsf{PKE.Valid}(pk, s)$ accepts, if $(y_1, w) \in R_2 \cap R_3$, then, $(y_1, \widetilde{w}) \in R_5$.

With a similar approach we can argue that

$$(y_1', w) \in R' \wedge ((y_1', x_2), \widetilde{w}) \in R_4 \Rightarrow ((y_1', x_2), \widetilde{w}) \in R_5.$$

This completes the proof. □

We will employ the relaxed ZAP scheme for the languages L and $\widetilde{L}$.

5.1 Construction

Construction 5.2. Let $\widetilde{D} = \widetilde{D}(\lambda, N)$ be the maximum depth of the NP verifier circuit for language $\widetilde{L}$ restricted to statements where the the ring has at most N members, and the security parameter corresponding to SPB hash keys and values and PKE ciphertext is λ. By $n = n(\lambda, \log N)$ denote the maximum size of the statements of language L where the ring has at most N members and the security parameter is λ. Recall that for security parameter λ, secret keys in PKE have size $\widetilde{\ell} = \ell_{sk}(\lambda)$. We now describe our ring signature construction:

- $\mathsf{Gen}(1^\lambda, N)$:
  - sample signing and verification keys $(vk, sk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$,
  - sample pk uniformly from the keyspace of PKE,
  - compute the first message $\rho \leftarrow \mathsf{ZAP.V}(1^\lambda, 1^n, 1^{\widetilde{\ell}}, 1^{\widetilde{D}})$ for the relaxed ZAP scheme,
  - output the verification key $\mathsf{VK} = (vk, pk, \rho)$ and signing key $\mathsf{SK} = (sk, vk, pk, \rho)$.
- $\mathsf{Sign}(\mathsf{SK}, m, \mathsf{R} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_\ell))$:
  - parse $\mathsf{SK} = (sk, vk, pk, \rho)$,
  - compute $\sigma \leftarrow \mathsf{Sig.Sign}(sk, m)$,
  - let $\mathsf{VK} = \mathsf{VK}_\mathsf{i} \in \mathsf{R}$ be the verification key corresponding to SK,
  - sample hash keys $(hk_1, shk_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, i)$, and compute the hash $h_1 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_1, \mathsf{R})$,
  - compute the opening $\tau_1 \leftarrow \mathsf{SPB.Open}(hk_1, shk_1, \mathsf{R}, i)$ to position i,
  - compute $c_1 \leftarrow \mathsf{PKE.Enc}(pk, (\sigma, vk); r_{c_1})$
  - sample hash keys $(hk_2, shk_2) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, i)$ and compute the hash $h_2 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_2, \mathsf{R})$,
  - sample c_2 randomly from the ciphertext space of PKE,
  - let $\mathsf{VK}_1 = (vk_1, pk_1, \rho_1)$ denote the lexicographically smallest member of R (as a string; note that this is necessarily unique).,
  - fix statement $x_1 = (m, c_1, c_2, hk_1, hk_2, h_1, h_2)$, witness $w = (vk, pk, i, \tau_1, \sigma, r_{c_1})$, and statement $x_2 = \mathsf{R}$,
  - Compute $\pi \leftarrow \mathsf{ZAP.P}(\rho_1, x = (x_1, x_2), w)$,
  - output $\Sigma = (c_1, hk_1, c_2, hk_2, \pi)$.
- $\mathsf{Verify}(\Sigma, m, \mathsf{R})$:
  - identify the lexicographically smallest verification key VK_1 in R,
  - compute $h'_1 = \mathsf{SPB.Hash}(hk_1, \mathsf{R})$,
  - compute $h'_2 = \mathsf{SPB.Hash}(hk_2, \mathsf{R})$,
  - fix $x_1 = (m, c_1, c_2, hk_1, hk_2, h'_1, h'_2)$, and $x_2 = \mathsf{R}$,
  - determine ρ_1 in VK_1,
  - compute and output $\mathsf{ZAP.Verify}(\rho_1, x, \pi)$.

Completeness of Construction 5.2 follows by the completeness of SPB and ZAP. For compactness, notice that $\widetilde{D}$ is upper-bounded by a polynomial in λ and $\log N$, and therefore, since Construction 4.3 is compact, Construction 5.2 is also compact.

5.2 Unforgeability

Here, we prove that our ring signature scheme possesses the unforgeability property as defined in Definition 2.12. The proof strategy is as follows: we leverage the selective non-witness adaptive statement soundness of ZAP to conclude that there must be a valid signature σ in the forgery attempt, and essentially try to obtain this signature with significantly high probability so that we can devise a reduction to the existential unforgeability of Sig.

Theorem 5.3. *Construction 5.2 is unforgeable, assuming* Sig *is* EUF−CMA *secure,* PKE *has injective key generation and pseudorandom public keys,* SPB *is somewhere perfectly binding, and* ZAP *satisfies selective non-witness adaptive statement soundness.*

Proof. We start by considering a PPT adversary $\mathcal{A}$ that participates in the unforgeability game. Let $Q = \mathrm{poly}(\lambda)$ be an upper bound on the number of key queries made by $\mathcal{A}$.

We proceed with a hybrid argument to set up our reduction to the unforgeability of Sig. Consider the following hybrids:

Hybrid H_0: This is just the standard unforgeability game. In particular, for all $i \in [Q]$, the challenger in the game generates pk_i by sampling an element uniformly from the keyspace of PKE.

Hybrid H_1: In this experiment, the only difference is that, the challenger first picks a uniformly random secret key sk_{PKE} for PKE, and then generates the corresponding public keys for the adversary using this, namely $pk_i \leftarrow \mathsf{PKE.GenWithKey}(sk_{\mathsf{PKE}})$, for all $i \in [Q]$. The challenger now stores sk_{PKE}.

Lemma 5.4. *Assuming* PKE *has pseudorandom public keys,* $H_0 \overset{c}{\approx} H_1$.

Proof. Let $\mathcal{A}$ be a PPT adversary attempting to distinguish H_0 and H_1. We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ having the same advantage against the pseudorandomness of public keys of PKE. Here, $\mathcal{A}'$ is either given $\{pk_i \leftarrow \mathsf{GenWithKey}(sk)\}_{i\in[Q]}$ for a sk chosen uniformly at random or $\{pk_i \leftarrow \mathsf{PK}_\lambda\}_{i\in[Q]}$. We define $\mathcal{A}'$ to proceed exactly as in H_0 but using the public keys that is given to it as input. Clearly, if pk_is are generated with a single uniformly chosen secret key, then, the view of $\mathcal{A}$ is identical to H_1, whereas, if pk_is are chosen uniformly at random, the view of $\mathcal{A}$ is identical to H_0. □

Now, we will proceed to show that unforgeability holds in H_1. Consider the adversary's forgery attempt ($\Sigma^* = (c_1^*, hk_1^*, c_2^*, hk_2^*, \pi^*)$,$m^*$,$\mathsf{R}^*$). Define x_1^* as the statement corresponding to Σ^* as $x_1^* = (m^*, c_1^*, c_2^*, hk_1^*, hk_2^*, h_1^*, h_2^*)$, where $h_1^* = \mathsf{SPB.Hash}(hk_1^*, \mathsf{R}^*)$ and $h_2^* = \mathsf{SPB.Hash}(hk_2^*, \mathsf{R}^*)$. Let $\mathsf{VK}_1^* = (vk_1^*, pk_1^*, \rho_1^*)$ be the lexicographically smallest verification key in R^*.

Our next step is to show that if π^* is a valid proof for $x^* = (x_1^*, x_2^* = \mathsf{R}^*)$ under ρ_1^*, then, with overwhelming probability, $x^* \notin \widetilde{L}$.

Lemma 5.5. *In H_1, assuming* ZAP *satisfies selective non-witness adaptive statement soundness, and* PKE *has injective key generation,*

$$\Pr[x^* \in \widetilde{L} \wedge \mathsf{ZAP.Verify}(\rho_1^*, x^*, \pi^*) \textit{ accepts }] = \mathrm{negl}(\lambda).$$

Proof. It is enough to show that for each $j \in [Q]$,

$$\Pr[x^* \in \widetilde{L} \wedge \mathsf{ZAP.Verify}(\rho_j, x^*, \pi^*) \text{ accepts }] = \mathrm{negl}(\lambda),$$

where ρ_j denotes the ZAP first message corresponding to the jth verification key VK_j generated in the game.

Let $\mathcal{A}$ be an adversary attempting to output a forgery such that $x^* \in \widetilde{L} \wedge \mathsf{ZAP.Verify}(\rho_j, x^*, \pi^*)$ accepts. We build an adversary $\mathcal{A}'$ against the selective non-witness adaptive statement soundness of ZAP for languages L and $\widetilde{L}$ with fixed non-witness $\widetilde{w} = sk_{\mathsf{PKE}}$. The algorithm $\mathcal{A}'$ proceeds as follows:

- on input ZAP first message $\widehat{\rho}$, it sets $\rho_j = \widehat{\rho}$ and then proceeds exactly as H_1.
- upon recieving the forgery attempt Σ^* from $\mathcal{A}$, it constructs the corresponding x^* and π^*, and outputs (x^*, π^*).

To finish the proof of this lemma, we observe that if $x^* \in \widetilde{L}$, then, except with negligible probability, $(x^*, sk_{\mathsf{PKE}}) \in \widetilde{R}$. This is because, if $x^* \in \widetilde{L}$, then, by definition of $\widetilde{L}$, there exists a non-witness $\widetilde{w}^*$ such that,

$$\forall (vk_i^*, pk_i^*, \rho_i^*) \in \mathsf{R}^* : \mathsf{PKE.Valid}(pk_i^*, \widetilde{w}^*) \text{ accepts},$$

and since PKE has injective key generation, it follows that except with negligible probability, $\widetilde{w}^* = sk_{\mathsf{PKE}}$.$\square$

In the next lemma we show that if $x^* \notin \widetilde{L}$, then, by decrypting c_1^* or c_2^*, we can find a forgery for Sig.

Lemma 5.6. *In H_1, assuming* Sig *is* EUF$-$CMA *secure,* PKE *has injective key generation, and* SPB *is somewhere perfectly binding,*

$$\Pr[x^* \notin \widetilde{L}] = \mathrm{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary attempting to output a forgery such that $x^* \notin \widetilde{L}$. We build an algorithm $\mathcal{A}'$ against the EUF$-$CMA security of Sig. The algorithm $\mathcal{A}'$ proceeds as follows,

- on input $\widehat{vk}$, first picks an index $j \leftarrow [Q]$ uniformly at random, and then sets $vk_j = \widehat{vk}$. It then proceeds as in H_1 to prepare the rest of the verification keys.
- when $\mathcal{A}$ sends a signing query, if it is using keys from a party other than the jth party, it proceeds as in H_1, otherwise, it uses the EUF$-$CMA game's signing oracle to obtain a signature for the jth party and then continues exactly as in H_1.
- if $\mathcal{A}$ tries to corrupt the jth party, $\mathcal{A}'$ aborts.
- upon recieving the forgery attempt Σ^* from $\mathcal{A}$, it decrypts c_1^* using sk_{PKE} to recover σ_1^*.
  If $\mathsf{Sig.Verify}(vk_j, m^*, \sigma_1^*)$ accepts, it sets $\widehat{\sigma} := \sigma_1^*$. Otherwise, it decrypts c_2^* with sk_{PKE} to recover σ_2^*, and sets $\widehat{\sigma} = \sigma_2^*$. It outputs $(m^*, \widehat{\sigma})$.

To finish the proof, we show that with probability at least

$$\frac{1}{Q}(\Pr[x^* \notin \widetilde{L}] - \mathrm{negl}(\lambda)),$$

$(m^*, \widehat{\sigma})$ is a valid forgery for key vk_j. Without loss of generality assume that

$$(m^*, c_1^*, hk_1^*, h_1^*, \mathsf{R}^*) \notin \widehat{L}.$$

Observe that due to the way H_1 generates the public keys and also by definition of h_1^*,

$$(m^*, c_1^*, hk_1^*, h_1^*, \mathsf{R}^*) \in L_4.$$

Therefore, by definition of $\widehat{L}$, there exists a string $\widetilde{w}$ such that,

$$((m^*, c_1^*, hk_1^*, h_1^*, \mathsf{R}^*), \widetilde{w}) \in R_5.$$

By an argument similar to the one presented in Lemma 5.5, it follows that except with negligible probability $\widetilde{w} = sk_{\mathsf{PKE}}$. Consequently, (i) $\mathsf{PKE.Dec}(sk_{PKE}, c_1^*) = (\sigma^*, vk^*)$ for some σ^* and vk^*, (ii) due to the somewhere perfectly binding property of SPB, there exists $\mathsf{VK}^* = (vk^*, pk^*, \rho^*)$ such that $\mathsf{VK}^* \in \mathsf{R}^*$, and finally (iii) $\mathsf{Sig.Verify}(vk^*, m^*, \sigma^*)$ accepts. We conclude that the adversary uses a verification key $\mathsf{VK}^* \in \mathsf{R}^*$ and that c_1^* encrypts (among other things) a signature σ^* that is valid for the forgery message m^* w.r.t. key vk^*. Since index j is chosen uniformly at random, $vk_j = vk^*$ with probability $1/Q$. □

Lemma 5.5 and Lemma 5.6 show that any efficient adversary has negligible chance of winning the $\mathsf{RS-FORGE}$ game in hybrid H_1. We observe that winning the $\mathsf{RS-FORGE}$ game is an event that can be efficiently tested, therefore, by Lemma 5.4 no efficient adversary can win the $\mathsf{RS-FORGE}$ game in hybrid H_0, i.e., Construction 5.2 is unforgeable. □

5.3 Anonymity

We now prove that our construction satisfies anonymity. Recall that this corresponds to an experiment where the adversary recieves the secret keys and randomness of all the existing parties, and then recieves a challenge signature created using the keys of one of two possible parties (of course, the challenge ring may also include parties that were created by the adversary). Our task is to show that the adversary cannot distinguish between a signature created by party i_0 and one created by party i_1 (for any distinct i_0, i_1). We will do this using a sequence of hybrids. Our strategy will be roughly as follows: we start with a signature produced using the signing key of party i_0. First, we switch c_2 to valid encryptions of a signature under vk_{i_1} (along with vk_{i_1}) and hk_2 to a valid SPB hash key to the index for VK_{i_1} in the ring respectively. Next, we switch the witness used in π to use these values (instead of c_1 and hk_1). Then, we change c_1 to valid encryption of a signature under vk_{i_1} and a valid SPB hash key to the index for VK_{i_1} in the ring respectively. Finally, we change c_2 to a junk ciphertext, as in the honest signing algorithm. The final hybrid just outputs a signature using the keys for party i_1, and thus we only have to show that the adversary cannot detect any of the individual changes outlined above.

Theorem 5.7. *Assume* PKE *has close to uniform ciphertexts as described in Definition 2.4,* SPB *is index hiding, and* ZAP *is statistically witness indistinguishable. Then the ring signature scheme in Construction 5.2 satisfies the anonymity property described in Definition 2.12.*

Proof. Let $\mathcal{A}$ be a PPT adversary participating in the anonymity game. Let $Q = \mathrm{poly}(\lambda)$ be an upper bound on the number of key queries made by $\mathcal{A}$. Suppose that the adversary's eventual challenge is $(\mathsf{R}, m, i_0, i_1)$. Let t_0, t_1 be the indices of VK_{i_0}, VK_{i_1} in R respectively. Denote by ρ the ZAP first message corresponding to the lexicographically smallest VK in R. As pointed out, it suffices to show that a signature prepared using SK_{i_0} is indistinguishable from one prepared using SK_{i_1}, even when $\mathcal{A}$ has all the keys $\mathsf{VK}_1, \cdots, \mathsf{VK}_Q$ and the randomness used in creating them. We do so using the following hybrids:

Hybrid H_0: This hybrid simply runs the anonymity game honestly as the challenger, and sends an honest signature generated using SK_{i_0}, namely $\Sigma = (c_1, hk_1, c_2, hk_2, \pi)$, as the challenge to the adversary.

Hybrid H_1: The only change in this hybrid is that it samples hk_2 in the signature with index t_1, i.e. $(hk_2, shk_2) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, t_1)$.

Hybrid H_2: The only difference between this hybrid and H_1 is that here, instead of sampling c_2 uniformly from the PKE ciphertext space, it generates c_2 as $c_2 \leftarrow \mathsf{PKE.Enc}(pk_{i_1}, (\sigma', vk_{i_1}); r_{c_2})$, where, $\sigma' \leftarrow \mathsf{Sig.Sign}(sk_{i_1}, m)$.

Hybrid H_3: This hybrid works exactly like the previous one, except that it uses a witness corresponding to (c_2, hk_2) to generate the proof π. Namely, it computes witness $w' = (vk_{i_2}, pk_{i_2}, t_1, \tau_2', \sigma', r_{c_2})$, where, $\tau_2' = \mathsf{SPB.Open}(hk_2, shk_2, \mathsf{R}, t_1)$, and proof π is generated as $\pi \leftarrow \mathsf{ZAP.P}(\rho, x, w')$.

Hybrid H_4: This hybrid is similar to H_3, except that it now computes c_1 by sampling it uniformly from the ciphertext space of PKE.

Hybrid H_5: This hybrid works exactly like the previous, with the only difference being that it generates hk_1 with respect to index t_1, i.e., $(hk_1, shk_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, t_1)$.

Hybrid H_6: It is identical to H_5, except that here, $c_1 \leftarrow \mathsf{PKE.Enc}(pk_{i_1}, (\sigma', vk_{i_1}); r_{c_1})$ where $\sigma' \leftarrow \mathsf{Sig.Sign}(sk_{i_1}, m)$.

Hybrid H_7: The only change in this hybrid is that, it uses a witness corresponding to (c_1, hk_1) to generate the ZAP proof. Namely, it computes $w'' = (vk, pk, t_1, \tau_1', \sigma, r_{c_1},)$, where $\tau_1' = \mathsf{SPB.Open}(hk_1, shk_1, \mathsf{R}, t_1)$, and $\pi = \mathsf{ZAP.P}(\rho, x, w'')$, and uses this in Σ.

Hybrid H_8: This hybrids works exactly like the previous, except that it now computes c_2 by sampling it uniformly from the ciphertext space of Enc. Notice that this hybrid corresponds to generating the signature using SK_{i_1}.

Lemma 5.8. *Assuming* SPB *is index hiding,* $H_0 \overset{c}{\approx} H_1$.

Proof. Let $\mathcal{A}$ be an adversary attempting to distinguish H_0 and H_1. We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ having the same advantage against the index hiding property of SPB. $\mathcal{A}'$ runs $\mathcal{A}$ and interacts with it exactly like H_0, till the point where $\mathcal{A}$ sends its challenge $(\mathsf{R}, m, i_0, i_1)$. At this point, $\mathcal{A}'$ sends $(t_0, t_1, |\mathsf{R}|)$ to its index hiding challenger. $\mathcal{A}'$ then receives a SPB hash key hk^*, which is either $\mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, t_0)$ or $\mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, t_1)$. It uses hk^* as the key hk_2 for generating the challenge signature Σ for $\mathcal{A}$. If hk^* is generated for index t_0 then $\mathcal{A}$'s view is identical to its view in H_0. Otherwise, if hk^* corresponds to t_1, $\mathcal{A}$'s view is identical to its view in H_1. □

Lemma 5.9. *If* PKE *has almost uniform ciphertexts,* $H_1 \overset{s}{\approx} H_2$.

Proof. This follows directly from the definition of almost uniform ciphertexts property described in Definition 2.4. □

Lemma 5.10. *If* ZAP *is statistically witness indistinguishable,* $H_2 \overset{s}{\approx} H_3$.

Proof. We observe that the witnesses w used in H_2 and w' used in H_3 are both valid ones for $x \in L$. This is the only change between these hybrids, and the lemma then follows almost immediately from the statistical witness indistinguishability of our ZAP. □

Lemma 5.11. *If* PKE *has close to uniform ciphertexts,* $H_3 \overset{s}{\approx} H_4$.

Proof. This follows directly from the definition of close to uniform ciphertexts property described in Definition 2.4. □

Lemma 5.12. *Assuming* SPB *is index hiding,* $H_4 \overset{c}{\approx} H_5$.

Proof. The proof of the lemma is almost identical to Lemma 5.8. □

Lemma 5.13. *If* PKE *has almost uniform ciphertexts,* $H_5 \overset{s}{\approx} H_6$.

Proof. This follows directly from the definition of close to uniform ciphertexts property described in Definition 2.4. □

Lemma 5.14. *If* ZAP *is statistically witness indistinguishable, and* PKE *has sparse valid public keys,* $H_6 \overset{s}{\approx} H_7$.

Proof. The proof for this lemma is very similar to Lemma 5.10 and we won't repeat it. □

Lemma 5.15. *If* PKE *has almost uniform ciphertexts,* $H_7 \overset{s}{\approx} H_8$.

Proof. This follows directly from the definition of close to uniform ciphertexts property described in Definition 2.4. □

This completes the proof of Theorem 5.7 □

6 Acknowledgments

We thank anonymous reviewers for pointing out issues in Definition 2.6 and Definition 3.1 in an earlier version of this work.

Omkant Pandey is supported in part by DARPA SIEVE Award HR00112020026, NSF grants 1907908 and 2028920, and a Cisco Research Award.

Sanjam Garg is supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation and Visa Inc.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, AFOSR, NSF, Cisco, Sloan Foundation, or Visa Inc.

References

[AOS02] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT*, pages 415–432. 2002. 1.4

[BD18] Z. Brakerski and N. Döttling. Two-message statistically sender-private OT from LWE. In *TCC*, pages 370–390. 2018. 1.3, 2.8

[BDF+11] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *International conference on the theory and application of cryptology and information security*, pages 41–69. Springer, 2011. A

[BDH+19] M. Backes, N. Döttling, L. Hanzlik, K. Kluczniak, and J. Schneider. Ring signatures: Logarithmic-size, no setup - from standard assumptions. In *EUROCRYPT*, pages 281–311. 2019. 1, 1.2, 1.3, 1.4, 2.3, 2.6, 2.10, 2.8, 2.12, 5

[BFJ+20] S. Badrinarayanan, R. Fernando, A. Jain, D. Khurana, and A. Sahai. Statistical ZAP arguments. In *EUROCRYPT*, pages 642–667. 2020. 1.1, 1.3

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432. 2003. 1.4

[BH18] X. Boyen and T. Haines. Forward-secure linkable ring signatures. In *ACISP*, pages 245–264. 2018. 1.4

[BHKS18] M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In *ASIACRYPT*, pages 405–434. 2018. 1.4

[BK10] Z. Brakerski and Y. T. Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086, 2010. `https://eprint.iacr.org/2010/086`. 1.4

[BKM06] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*, pages 60–79. 2006. 1.3, 1.4, 2.8

[BKP20] W. Beullens, S. Katsumata, and F. Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYP*, pages 464–492. 2020. 1.4

[BLO18] C. Baum, H. Lin, and S. Oechsner. Towards practical lattice-based one-time linkable ring signatures. In *ICICS*, pages 303–322. 2018. 1, 1.4

[BLP+13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584. 2013. 2.1

[Blu87] M. Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451. 1987. 2.4, 2.4

[BZ13a] D. Boneh and M. Zhandry. Quantum-secure message authentication codes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 592–608. Springer, 2013. A

[BZ13b] D. Boneh and M. Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *Annual cryptology conference*, pages 361–379. Springer, 2013. A, A.1, A.1, A.2, A.2, A.2, A.2

[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 *STOC*. 2019. To appear. 1.2, 1.3

[CGS07] N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In *ICALP*, pages 423–434. 2007. 1.4

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010. 2.7

[CWLY06] S. S. M. Chow, V. K. Wei, J. K. Liu, and T. H. Yuen. Ring signatures without random oracles. In *ASIACCS*, pages 297–302. 2006. 1.4

[DKNS04] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT*, pages 609–626. 2004. 1.4

[DN00] C. Dwork and M. Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. 1.3

[EZS+19] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *CCS*, pages 567–584. 2019. 1.4

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194. 1986. 1.2

[Gen09] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`. 1.3

[Gha13] E. Ghadafi. Sub-linear blind ring signatures without random oracles. In *IMACC*, pages 304–323. 2013. 1.4

[GJJM20] V. Goyal, A. Jain, Z. Jin, and G. Malavolta. Statistical zaps and new oblivious transfer protocols. In *EUROCRYPT*, pages 668–699. 2020. 1.1, 1.3, 4.2

[GK15] J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, pages 253–280. 2015. 1.4

[Gon17] A. González. A ring signature of size Θ(sqrt[3]{n}) without random oracles. 2017. `https://eprint.iacr.org/2017/905`. 1.4

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008. 1

[HS03] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In *INDOCRYPT*, pages 266–279. 2003. 1.4

[KKW18] J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 525–537. 2018. 1.4

[LLNW16] B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT*, pages 1–31. 2016. 1.4

[LNPY20] B. Libert, K. Nguyen, T. Peters, and M. Yung. One-shot fiat-shamir-based nizk arguments of composite residuosity in the standard model. Cryptology ePrint Archive, Report 2020/1334, 2020. `https://eprint.iacr.org/2020/1334`. 1.4

[LNS21] V. Lyubashevsky, N. K. Nguyen, and G. Seiler. SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. In *CRYPTO*. 2021. To appear. 1.4

[LPQ18] B. Libert, T. Peters, and C. Qian. Logarithmic-size ring signatures with tight security from the DDH assumption. In *ESORICS*, pages 288–308. 2018. 1, 1.4

[LVW19] A. Lombardi, V. Vaikuntanathan, and D. Wichs. 2-message publicly verifiable WI from (subexponential) LWE. *Cryptology ePrint Archive, Report 2019/808*, page 808, 2019. 1.1, 1.3

[Mer87] R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378. 1987. 1.2

[MS17] G. Malavolta and D. Schröder. Efficient ring signatures in the standard model. In *ASIACRYPT*, Lecture Notes in Computer Science, pages 128–157. 2017. 1.4

[Noe15] S. Noether. Ring signature confidential transactions for monero. 2015. `https://eprint.iacr.org/2015/1098`. 1

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. 1990. 1.2

[OPP14] R. Ostrovsky, A. Paskin-Cherniavsky, and B. Paskin-Cherniavsky. Maliciously circuit-private FHE. In *CRYPTO*, pages 536–553. 2014. 1.3, 2.7, 2.8, 3.2, 3.2

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009. 2.1

[PRS17] C. Peikert, O. Regev, and N. Stephens-Davidowitz. Pseudorandomness of Ring-LWE for any ring and modulus. In *STOC*, pages 461–473. 2017. 2.1

[PS96] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 252–265. Springer, 1996. A.1

[PS19a] S. Park and A. Sealfon. It wasn't me! - repudiability and claimability of ring signatures. In *CRYPTO*, pages 159–190. 2019. 1, 1.4

[PS19b] C. Peikert and S. Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, pages 89–114. 2019. 1.2, 1.3, 2.2, 2.2, 2.3

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005. 1.3, 2.1, 2.3, 1

[RT01] R. Rivest and A. S. Y. Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565. 2001. 1, 1.4

[SS10] S. Schäge and J. Schwenk. A cdh-based ring signature scheme with short signatures and public keys. In *Financial Cryptography and Data Security*, pages 129–142. 2010. 1.4

[SW07] H. Shacham and B. Waters. Efficient ring signatures without random oracles. In *PKC*, pages 166–180. 2007. 1.4

[TSS+18] W. A. A. Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1. 0). In *Australasian Conference on Information Security and Privacy*, pages 558–576. Springer, 2018. 1, 1.4

[Zha12] M. Zhandry. Secure identity-based encryption in the quantum random oracle model. In *Annual Cryptology Conference*, pages 758–775. Springer, 2012. A

A Quantum Security for Ring Signatures

In the literature for post-quantum signature schemes, one encounters two flavours of security definitions against quantum adversaries.

- **Weak Security:** Here the security games are defined as in the classical case, except that the adversary is allowed to be a QPT machine. We emphasize that the communication between the challenger and the adversary is still classical. Namely, the adversary is not allowed to query the challenger's circuit in a quantum manner. The technical challenges in these settings typically stem from schemes in the quantum random oracle model (where the random oracle can be queried in superposition), see e.g., [BDF+11, Zha12].
- **Full Security:** In this model we allow the adversary to query the challenger in a quantum manner. Even defining the right notion of security in these settings can be non-trivial [BZ13b, BZ13a], and the exact definition depends closely on the functionality that one requires.

In this section we lift the definition of anonymity and unforgeability to the quantum settings (in the full sense), then we present a generic transformation that allows us to compile any weakly post-quantum secure ring signature scheme into a fully quantum secure one (under our definitions).

A.1 Defining (Strong) Quantum Security

We present the definitions of anonymity and unforgeability below.

Quantum Anonymity. Recall that in the classical anonymity game (Definition 2.12), the adversary sends i_0 and i_1 as her challenge identities, and gets a signature signed using the key of identity i_b (w.r.t. to the R and m picked by her). Towards defining full quantum security, a natural first attempt is to allow the adversary to send a superposition of the form $\sum \psi_{i_0,i_1,\mathsf{R},m}|i_0, i_1, \mathsf{R}, m\rangle$, and let the challenger sign using the key for the i_b register in all the positions. In the following, we argue that it may not be a good idea to allow superpositions over the challenge identities.

On quantum challenge identities. We show that allowing the adversary to issue queries on superpositions of identities leads to an unsatisfiable definition. This is due essentially to the same reason as in [BZ13b, Thm. 4.2], where the authors show that allowing quantum *challenge* query will lead to an unsatisfiable IND-CPA notion for encryptions. Using the same technique, we can construct an adversary that breaks the anonymity by means of a single quantum query with superposition over the identities. More specifically, the challenger will answer the adversary's quantum query in the following way:

$$\sum \psi_{i_0,i_1,\mathsf{R},m,t}|i_0, i_1, \mathsf{R}, m, t\rangle \rightarrow \sum \psi_{i_0,i_1,\mathsf{R},m,t}|i_0, i_1, \mathsf{R}, m, t \oplus \mathsf{Sign}(\mathsf{SK}_{i_b}, m, \mathsf{R}; r)\rangle.$$

The adversary puts a uniform superposition of all valid identities in the first register, and a single, *classical* identity in the second registers. She uses arbitrary but *classical* m and R, as long as they are valid. She also sets t to be a *classical* string. After the challenger's signing operation, observe that if $b = 0$, the last register will contain signatures in superposition (as i_0 is in superposition); if $b = 1$, it will contain a classical signature (as i_1 is classical). The adversary then measures the last register. If $b = 0$, the resulting state (over all the 5 registers) will be a purely classical state; if $b = 1$, the measurement does nothing, so the first register still contains a superposition over all valid identities. These two cases can be efficiently distinguished by means of a Fourier transform on the first register followed by a measurement. Therefore, to obtain a meaningful notion of quantum anonymity, we require the adversary to send its challenge identities (i_0, i_1) in classical form.

Superposition on the ring. Due to technical reasons, we do not know how to prove quantum security of our construction if we allow the superposition attacks on the ring. Looking ahead, we will handle the superposition over the message by hashing it using pair-wise independent hash functions. This technique (from [BZ13b]) allows us to replace exponentially many messages (in superposition) with polynomially many random strings. But this does not work for rings (in the sense of ring signatures)—hashing will destroy the structure of the public keys in the ring. For this reason, in our definition, we will ask the adversary to send the challenge ring in classical manner.

We stress that this choice should not be attributed to definitional issues; it really is due to the restriction of our techniques. Unlike the case for challenge identities, a superposition on the ring does *not* mean the adversary asks a superposition of the members to sign. Instead, the signer is defined by the signing key (the identity). The superpositions over the ring have a similar interpretation to those over messages—both the ring and the message are simply inputs to the signer's algorithm; superposition attacks on them are possible if the adversary manages to trick the signer into showing quantum behavior. We leave it as an open question to achieve security against attacks employing a superposition on both the ring and the message.

The above discussion leads to the following definition for anonymity:

Definition A.1 (Quantum Anonymity). Consider a triple of PPT algorithms $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies the same syntax as in Definition 2.12. RS achieves quantum anonymity if, for any QPT algorithm $\mathcal{A}$ and any $q = \mathrm{poly}(\lambda)$, $\mathcal{A}$'s probability of success in the following game is negligible in λ:

1. For each $i \in [q]$, the challenger generates key pairs $(\mathsf{VK}_i, \mathsf{SK}_i) \leftarrow \mathsf{RS.Gen}(1^\lambda; r_i)$ (where r_i denotes the randomness used to generate the i th key pair). This defines a set $\{\mathsf{VK}_i\}_{i\in[q]}$. The challenger sends $\{(\mathsf{VK}_i, \mathsf{SK}_i, r_i)\}_{i\in[q]}$ to $\mathcal{A}$;

2. $\mathcal{A}$ sends (i_0, i_1) to the challenger, where both i_0 and i_1 are in $[q]$;

3. $\mathcal{A}$'s challenge query is allowed to be on superpositions of messages. The challenger picks a random bit b and a random string r. It signs the message using SK_{i_b} and randomness r, while making sure that VK_{i_0} and VK_{i_1} are indeed in the ring specified by $\mathcal{A}$. Formally, the challenger implements the following mapping:
$$\sum_{m^*,t} \psi_{m^*,t}|m^*, t\rangle \to \sum_{m^*,t} \psi_{m^*,t}|m^*, t \oplus f(m^*)\rangle,$$
where $f(m^*) = \begin{cases} \mathsf{RS.Sign}(\mathsf{SK}_{i_b}, m^*, R^*; r) & \text{if } \mathsf{VK}_{i_0}, \mathsf{VK}_{i_1} \in R^* \\ \perp & \text{otherwise} \end{cases}$.

4. $\mathcal{A}$ sends back its guess bit b'. The challenger outputs 1 if $b' = b$, otherwise it outputs 0.

Quantum Unforgeability. In the unforgeability game, the adversary can make corruption queries. Just as in the case of challenge identities, we require such corruption queries to be *classical*, as it does not make much sense to "corrupt a superposition of all the ring members". The adversary also makes signing queries. Ideally, we should now give her quantum access to the signing oracle. But for the same reason as before, we do not know how to handle superpositions on the ring. Thus, we will only allow the adversary to use superpositions on the message.

On quantum forgeries. It also requires extra caution to define "valid" forgeries in the quantum setting. Each quantum signing query can be a superposition of every message in the message space. Sampling the returned superposition will result in a single message/signature pair for a random message. Therefore, the classical notion of existential forgery being a signature on a new message is ill-defined. This issue already appeared in [BZ13b] when the authors tried to define unforgeability for ordinary signatures. There, the authors require that the adversary cannot produce $\mathsf{sq}+1$ valid message/signature pairs with sq signing queries (an approach previously used for blind signatures [PS96]). When restricted to the classical setting, this definition is equivalent to the standard unforgeability of ordinary signatures.

We want to adapt the above approach in the ring signature setting. But note that the adversary's forgery now contains a ring R^*. This forgery will be considered as valid only if R^* consists of the uncorrupted members (i.e. $\mathsf{R}^* \subseteq \mathcal{VK} \setminus \mathcal{C}$). We can generalize the approach in [BZ13b] by requiring that, with sq quantum signing queries, adversary cannot produce $\mathsf{sq}+1$ signatures, where all the rings are subsets of $\mathcal{VK} \setminus \mathcal{C}$. We remark that this definition, when restricted to the classical setting, is weaker than the standard unforgeability in Definition 2.12. But it serves as a good starting point for quantum unforgeability of ring signatures. It will be interesting for future research to explore this definitional issue.

Based on the above discussion, we present our definition for quantum unforgeability as follows.

Definition A.2 (Quantum Unforgeability). Consider a triple of PPT algorithms $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies the same syntax as in Definition 2.12. RS achieves quantum unforgeability if, for any QPT algorithm $\mathcal{A}$ and any $q = \mathrm{poly}(\lambda)$, $\mathcal{A}$'s probability of success in the following game is negligible in λ:

1. For each $i \in [q]$, the challenger generates key pairs $(\mathsf{VK}_i, \mathsf{SK}_i) \leftarrow \mathsf{RS.Gen}(1^\lambda; r_i)$ using randomness r_i. It then defines $\mathcal{VK} = \{\mathsf{VK}_i\}_{i\in[q]}$ and initializes a set $\mathcal{C} = \emptyset$.
2. The challenger sends $\mathcal{VK}$ to $\mathcal{A}$.
3. $\mathcal{A}$ is allowed make the following two kinds of queries:
   - *Corruption Queries:* Upon receiving a (classical) query of the form $(\mathsf{corrupt}, i)$: the challenger adds VK_i to the set $\mathcal{C}$ and returns the randomness r_i to $\mathcal{A}$.
   - *Quantum Signing Queries:* $\mathcal{A}$ is allowed to query the signing oracle on some index i, some ring R, and superpositions over messages. The challenger samples a random string r and computes the following mapping:
   $$\sum_{m,t} \psi_{m,t}|m,t\rangle \rightarrow \sum_{m,t} \psi_{m,t}|m,t \oplus f(m,i)\rangle,$$
   where $f(m,i) = \begin{cases} \mathsf{RS.Sign}(\mathsf{SK}_i, m, \mathsf{R}; r) & \text{if } \mathsf{VK}_i \in \mathsf{R} \\ \perp & \text{otherwise} \end{cases}$.
4. Assume that $\mathcal{A}$ made sq quantum signing queries in total. $\mathcal{A}$ is required to produce $\mathsf{sq}+1$ tuples of the form $\{(\mathsf{R}_j^*, m_j^*, \Sigma_j^*)\}_{j\in[\mathsf{sq}+1]}$. The challenger outputs 1 if for all $j \in [\mathsf{sq}+1]$ it holds that $R_j^* \subseteq \mathcal{VK}\setminus\mathcal{C}$ and $\mathsf{RS.Verify}(\mathsf{R}_j^*, m_j^*, \Sigma_j^*) = 1$.

A.2 Our Transformation

In the following we preset our quantum-secure ring signature scheme in the form of a generic transformation: Given a weakly secure scheme (where the adversary has classical access to all oracles) we compile it into a fully secure one.

Preliminaries. Before delving into the description of the construction we recall some useful lemmas from [BZ13b].

Lemma A.3. *Let A be a quantum algorithm, and $\Pr[x]$ be the probability that A outputs x. Let A' be another quantum algorithm obtained from A by pausing A at an arbitrary stage of execution, performing a partial measurement that obtains one of k outcomes, and then resuming A. Let $\Pr'[x]$ be the probability that A' outputs x. Then $\Pr'[x] \geq \Pr[x]/k$.*

Lemma A.4. *Let H be an oracle drawn from a $2q$-wise independent distribution. Then the advantage of any quantum algorithm making at most q queries to H has in distinguishing H from a truly random function, is identically 0.*

We next define the notion of small-range distributions, and recall some more associated lemmata from [BZ13b].

Definition A.5 (Small-range distributions). Fix sets $\mathcal{X}$ and $\mathcal{Y}$ and a distribution D on $\mathcal{Y}$. Fix an integer r. Let $\mathbf{y} = (y_1, \ldots, y_r)$ be a list of r samples from D and let P be a random function from $\mathcal{X}$ to $[r]$. The distributions on $\mathbf{y}$ and P induce a distribution on functions $H : \mathcal{X} \to \mathcal{Y}$ defined by $H(x) = y_{P(x)}$. This distribution is called a *small-range distribution* D on $\mathcal{Y}$.

Lemma A.6. *There is a universal constant C_0 such that, for any sets $\mathcal{X}$ and $\mathcal{Y}$, distribution D on $\mathcal{Y}$, any integer ℓ, and any quantum algorithm A making q queries to an oracle $H : \mathcal{X} \to \mathcal{Y}$, the following two cases are indistinguishable, except with probability less than $C_0 q^3/\ell$:*

- *$H(x) = y_x$ where $\mathbf{y}$ is a list of samples from D of size $|\mathcal{X}|$.*
- *H is drawn from the small-range distribution with ℓ samples from D.*

Lemma A.7. *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, and for each $x \in \mathcal{X}$, let D_x and D'_x be distributions on $\mathcal{Y}$ such that $|D_x - D'_x| \leq \epsilon$ for some value ϵ that is independent of x. Let $O : \mathcal{X} \to \mathcal{Y}$ be a function where, for each x, $O(x)$ is drawn from D_x, and let $O'(x)$ be a function where, for each x, $O'(x)$ is drawn from $D'(x)$. Then any quantum algorithm making at most q queries to either O or O' cannot distinguish the two, except with probability at most $\sqrt{8C_0 q^3 \epsilon}$.*

We also recall the notion of chameleon hash, as defined in [BZ13b].

Definition A.8 (Chameleon Hash Functions). A chameleon hash function $\mathcal{H}$ is a tuple of efficient algorithms $(\mathsf{HGen}, \mathsf{H}, \mathsf{Inv}, \mathsf{Sample})$ where:

- $\mathsf{HGen}(\lambda)$ generates a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{H}(\mathsf{pk}, m, r)$ maps messages to space $\mathcal{Y}$.

- $\mathsf{Sample}(\lambda)$ samples r from some distribution such that, for every pk and m, $\mathsf{H}(\mathsf{pk}, m, r)$ is uniformly distributed.
- $\mathsf{Inv}(\mathsf{sk}, h, m)$ produces an r such that $\mathsf{H}(\mathsf{pk}, m, r) = h$, and r is distributed negligibly-close to $\mathsf{Sample}(\lambda)$ conditioned on $\mathsf{H}(\mathsf{pk}, m, r) = h$.

A chameleon hash function is said to be *collision resistant* if no efficient quantum algorithm that is given pk can produce collisions in $\mathsf{H}(\mathsf{pk}, \cdot, \cdot)$

Construction. On input a ring signature $\mathsf{RS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, we define the augmented scheme analogously, except with the following modifications:

- In the Gen algorithm, we additionally sample a key for a chameleon hash function $(s, k) \leftarrow \mathsf{HGen}(1^\lambda)$.
- In the Sign algorithm, we sample two pair-wise independent functions P and Q, then we compute $r \leftarrow \mathsf{Sample}(1^\lambda; P(m \| \mathsf{R}))$ and $h \leftarrow \mathsf{Hash}(k, m \| \mathsf{R}, r)$. Finally, we output the signature $\Sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, h, R; Q(m \| \mathsf{R}))$ and the randomness r.
- The Verify algorithm uses r to recompute h, then verify the signature Σ using the original verification algorithm.

Where $\mathsf{H} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{Sample}, \mathsf{Inv})$ is a chameleon hash as defined above.

Unforgeability. It was shown in [BZ13b] that any signature scheme can be lifted to the fully quantum settings with the same transformation described above. While this was not formally shown for the case of ring signatures, the argument is essentially identical and we report the analysis here only for the sake of completeness.

Theorem A.9 (Unforgeability). *Let* RS *be a weakly secure ring signature scheme and let* H *be a chameleon hash, then the construction as described above is fully quantum unforgeable.*

Proof. We define the following series of hybrids.

Hybrid H_0: This is the original experiment.

Hybrid H_1: In this hybrid we switch the pair-wise independent functions to truly random. The view of the adversary is identical by Lemma A.3

Hybrid H_2: This is identical to the previous hybrid, except that we add the condition that none of the hashes in the set output by the adversary form a collision. The success probability is unchanged (up to a negligible function) by a reduction against the collision resistance of H.

Hybrid H_3: For all signing queries, we sample the value h uniformly and compute $r \leftarrow \mathsf{Inv}(s, h, m \| \mathsf{R}, t)$, where t is uniformly chosen. By Lemma A.7, the modified oracle is computationally indistinguishable from the eyes of the adversary.

Hybrid H_4: Let $\ell = 2 \cdot C_0 \cdot \mathsf{sq} \cdot p$ where p is a lower bound on the success probability of the attacker and C_0 is the constant from Lemma A.6. For all $i \in [\mathsf{sq}]$ and $j \in [\ell]$ sample a uniform $h_j^{(i)}$ and a uniform mapping O_i from the message space to $[\ell]$. Then for all queries define the oracle that maps the message pair to some hash in the set $H^{(i)} = \bigcup_j h_j^{(i)}$ and sign the corresponding $h_j^{(i)}$, using the input ring. The difference with respect to the previous hybrid is that each query is answered signing hashes from a small range distribution. By Lemma A.6, this modification is indistinguishable to the eyes of the adversary.

Hybrid H_5: Sample the mappings O_i and the randomnesses t using two pair-wise independent hash functions O_i and T_i. By Lemma A.4 the view of the adversary is unchanged.

Hybrid H_6: Consider two cases: If one of the hashes in the forgery set of the adversary does not belong to the total small-domain set of hashes $\bigcup_i H^{(i)}$ sampled by the challenger, then we have a forgery and therefore a contradiction to the unforgeability of the ring signature scheme. Otherwise, by the pigeonhole principle, it must be the case that two adversarial hashes belong to the same small domain subset $H^{(i)}$, for some $i \in [\mathsf{sq}]$.

Hybrid H_7: Guess a query i^* and abort if $i^* \neq i$, where i is the index for which the above condition is satisfied. Since the adversary asks only a polynomial amount of signing queries, the success probability is still a polynomial.

Hybrid H_8: For the i^* query, measure the output of O_{i^*} that determines which hash we need to sign. By Lemma A.3, the forgery happens with high enough probability. Note that here we just need to sign one hash, so there is some hash in the forgery set returned by the adversary that is never signed.

The proof is concluded with a straightforward reduction against the (weak) unforgeability of the ring signture scheme. □

Anonymity. In the following we show that the compiled ring signature construction is fully quantum anonymous.

Theorem A.10 (Anonymity). *Let* RS *be a weakly secure ring signature scheme and let* H *be a chameleon hash, then the construction as described above is fully quantum anonymous.*

Proof. We define the following series of hybrids.

Hybrid H_0: Fix the oracle that signs using SK_{i_0}.

Hybrid H_1: Same as H_1 in the proof of Theorem A.9.

Hybrid H_2: Same as H_3 in the proof of Theorem A.9.

Hybrid H_3: Same as H_4 in the proof of Theorem A.9, except that now we have only a single query to the signing oracle.

Hybrid H_4: Same as H_5 in the proof of Theorem A.9.

Hybrid H_5: Observe that now the oracle is efficiently simulatable and signs only a polynomial-size set of messages. We now switch the signing key used to sign the messages to SK_{i_1}. Indistinguishability follows by a standard hybrid argument assuming the (weak) anonymity of the scheme RS.

Hybrid H_5: Undo the changes done in H_4.

Hybrid H_6: Undo the changes done in H_3.

Hybrid H_7: Undo the changes done in H_2.

Hybrid H_8: Undo the changes done in H_1.

Observe that H_8 is the original game, except that the oracle uses the key SK_{i_1} to sign. This concludes our proof. □