Ring Signatures: Logarithmic-Size, No Setup — from Standard Assumptions

Michael Backes[1], Nico Döttling[1], Lucjan Hanzlik[1,2], Kamil Kluczniak[1], and Jonas Schneider[1]

[1] CISPA Helmholtz Center for Information Security
Saarland Informatics Campus
{backes,doettling,hanzlik,kamil.kluczniak,jonas.schneider}@cispa.saarland
[2] Stanford University
lucjan.hanzlik@stanford.edu

Abstract. Ring signatures allow for creating signatures on behalf of an ad hoc group of signers, hiding the true identity of the signer among the group. A natural goal is to construct a ring signature scheme for which the signature size is short in the number of ring members. Moreover, such a construction should not rely on a trusted setup and be proven secure under falsifiable standard assumptions. Despite many years of research this question is still open.
In this paper, we present the first construction of logarithmic-size ring signatures which do not rely on a trusted setup or the random oracle heuristic. Specifically, our scheme can be instantiated from standard assumptions and the size of signatures grows only logarithmically in the number of ring members.
We also extend our techniques to the setting of linkable ring signatures, where signatures created using the same signing key can be linked.

Keywords: ring signatures, linkable ring signatures, standard model

1 Introduction

Ring signatures, introduced by Rivest, Shamir and Tauman-Kalai [34] allow a signer to hide in a crowd, or *ring* of potential signers. More specifically, the signing algorithm of a ring signature scheme takes as additional input a list of verification keys R and outputs a signature. Such a signature can be verified given the ring R. The feature of interest of ring signatures is that given such a signature, no one, not even an insider in possession of all the secret keys corresponding to the verification keys in the ring, can tell which key was used to compute this signature. The original motivation for ring signatures was whistleblowing, where the leaking party can hide her identity and at the same time convince outsiders that the leaked information is genuine (by using a ring composed only of people with access to this information). In terms of security two properties are required of ring signatures: unforgeability and anonymity. The first property requires that an efficient adversary should not be able to forge a signature on behalf

of an honest ring of signers. Anonymity requires that signatures do not give away by which member they were created. This can be cast as an experiment in which the adversary has to guess which one out of two ring members created a signature.

The notion of linkable ring signatures [28] is an extension of the concept of ring signatures such that there is a public way of determining whether two signatures have been produced by the same signer. Linkable ring signatures yield a very elegant approach to e-voting [39]: Every voter is registered with their verification key. To cast a vote, all a voter has to do is to sign his or her vote on behalf of the ring of all registered voters. Linkability prevents voters from casting multiple votes. This can even be turned into an augmentation of the voting functionality by allowing voters to revote, where only the most recently cast votes of a set of votes that link counts.

Recently, linkable ring signature have also drawn attention in the domain of decentralized currencies, where they can be used to implement a mechanism for anonymized transactions. Linkable ring signatures are, for instance, used in a cryptocurrency called *Monero* [31], where they allow payers to hide their identity in an anonymity set composed of identities from previous transactions. Currently Monero uses a setup-free Schnorr based ring signature scheme [36] where the size of signatures scales linearly in the size of the ring. To decrease the size of the transaction by default Monero uses small rings, which provide only a limited amount of anonymity. The anonymity definition for linkable ring signatures needs to be different from the definition for standard ring signatures. We will elaborate further on this topic below. In both of the above applications two aspects are of the essence:

- The ring signature scheme should not rely on a trusted setup. Especially in the e-voting application it is of paramount importance for the acceptance of such a system that there *cannot exist* a trapdoor that enables deanonymization of voters.
- For practical purposes, e.g. for elections with millions of voters, the size of individual signatures should be essentially independent of the size of the ring of signers.

1.1 Our Contributions

In this work, we provide the first construction of ring signatures which simultaneously

- does not rely on a trusted setup or the random oracle heuristic,
- can be proven secure under *falsifiable* standard assumptions, namely the existence of non-interactive witness indistinguishable proofs [16, 3, 24, 7] and additional standard assumptions such as the hardness of the *Decisional Diffie Hellman* problem [17] or the *Learning with Errors* problem [33],
- has signatures of size $\log(\ell) \cdot \mathsf{poly}(\lambda)$, where ℓ is the size of the ring of signers and λ the security parameter.

Our work therefore settles the problem of logarithmic-size ring signatures in the standard model, which has been a long standing open problem. Furthermore, we extend our techniques to the domain of linkable ring signatures, i.e. we construct linkable ring signatures of size $\log(\ell) \cdot \mathsf{poly}(\lambda)$ without setup and in the standard model. Along the way, we introduce new techniques that enable us to use NIWI proofs instead of NIZK proofs, which may be of independent interest.

As an additional contribution, we propose a stronger security model for linkable ring signatures and prove that our linkable ring signature scheme is secure in this model.

1.2 Our Techniques

To describe our scheme, it is instructive to recall the standard model ring signature scheme of Bender, Katz, and Morselli [5]. In the BKM scheme, a verification key $\mathrm{VK} = (\mathsf{vk}, \mathsf{pk})$ consists of a verification key vk for a standard signature scheme and a public key pk for a public key encryption scheme. To sign a message m given a signing key sk and a ring $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$, one proceeds as follows. In a first step, locate verification key $\mathrm{VK}_{i^*} = (\mathsf{vk}_{i^*}, \mathsf{pk}_{i^*})$ corresponding to the signing key sk in the ring R. Now compute a signature σ of m using the signing key sk and encrypt σ under pk_{i^*} to obtain a ciphertext ct_{i^*}. Next, for all $i \neq i^*$ compute *filler ciphertexts* ct_i as encryptions of 0^λ under pk_i, where $\mathrm{VK}_i = (\mathsf{vk}_i, \mathsf{pk}_i)$. Finally, use a non-interactive[3] witness-indistinguishable proof π for the statement $(\mathsf{m}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ to show that there exists an index i^* such that ct_{i^*} encrypts a signature σ and that σ verifies for the message m under the verification key vk_{i^*}. The ring signature is now given by $\Sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$. To verify a signature Σ for a message m and ring R, use the NIWI verifier to verify that π is a proof for the statement $(\mathsf{m}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$.

We also briefly review how unforgeability and anonymity of this scheme are established. To establish unforgeability, note that by the perfect soundness of the NIWI proof π one of the ct_i must actually be an encryption of a signature of m under vk_i. The security reduction can therefore set up all the pk_i such that it knows the corresponding secret keys and decrypt the signature. Establishing anonymity relies on witness indistinguishability of the NIWI proof system. That is, the reduction can set up the signature Σ such that in fact two different ciphertexts ct_{i_0} and ct_{i_1} encrypt a valid signature (each under their corresponding verification key). We can now use witness indistinguishability to switch the witness from index i_0 to i_1. Thus we can establish that signatures computed using sk_{i_0} are computationally indistinguishable from signatures computed using sk_{i_1}. The size of the signature is linear in the ring size ℓ. There are two major obstacles in making the size of the signatures sublinear:

1. The signature contains all the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$.

[3] Bender et al. [5] actually use 2-message public-coin witness-indistinguishable proofs (ZAPs) rather than NIWI proofs, which is a slightly weaker primitive than NIWI proofs.

2. The witness for the validity of statement $(\mathsf{m}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ is also of size linear in ℓ.

Reducing the number of Ciphertexts. Starting from the BKM scheme, our first idea is that if we use an appropriate public key encryption scheme PKE, then we do not need to include all the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ in the signature, but only two ciphertexts ct and ct'. The additional property we need from PKE is that a ciphertexts ct cannot be linked to the public key pk that was used to compute ct, *unless* one is in the possession of the corresponding secret key sk. This property immediately holds if the public key encryption scheme PKE has pseudorandom ciphertexts. In fact, many constructions of public key encryption have pseudorandom ciphertexts, e.g. the classic ElGamal scheme based on DDH [17] or Regev's scheme based on LWE [33].

Our first modification is thus to compute ct by encrypting the signature σ under pk_{i^*} and choosing ct' uniformly at random. We also compute the proof π differently. Namely, we prove that for a statement of the form $(\mathsf{m}, \mathsf{ct}, \mathsf{ct}', \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ it holds that there exist indices i^* and $i^\dagger$ such that either ct is an encryption of a signature σ^* of m with respect to the verification key vk_{i^*} under the public key pk_{i^*}, *or* ct' is an encryption of a signature $\sigma^\dagger$ of m with respect to the verification key $\mathsf{vk}_{i^\dagger}$ under the public key $\mathsf{pk}_{i^\dagger}$. In this modified scheme, a signature $\Sigma = (\mathsf{ct}, \mathsf{ct}', \pi)$ consists of the two ciphertexts $\mathsf{ct}, \mathsf{ct}'$ and the proof π. Verification checks that π is a proof for the statement $(\mathsf{m}, \mathsf{ct}, \mathsf{ct}', \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$. We will briefly argue that this scheme is still unforgeable and anonymous. First observe that if the proof π for the statement $(\mathsf{m}, \mathsf{ct}, \mathsf{ct}', \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ verifies, then by the perfect soundness of the NIWI proof system either ct or ct' must encrypt a signature under a public key pk_{i^*} or $\mathsf{pk}_{i^\dagger}$ respectively. Therefore, we can again construct a reduction which knows all the secret keys corresponding to the pk_i. This way, the reduction will be able to decrypt the signature σ from ct or ct'. To show anonymity, we transform a signature computed with sk_{i_0} into a signature computed with sk_{i_1} via a sequence of hybrids. In the first hybrid step we will make ct' an encryption of a signature σ_1 of m with respect to the key vk_{i_1} under the public key pk_1. This change is possible as the ciphertexts of PKE are pseudorandom. Next, we will use witness-indistinguishability of NIWI to switch the witness for the statement $(\mathsf{m}, \mathsf{ct}, \mathsf{ct}', \mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$. The new witness shows that ct' encrypts a valid signature of m. This means that we do not need a witness for ct anymore. Thus, in the next hybrid steps, we replace the ciphertext ct by a random string, and then replace this random string by an encryption of the signature σ_1 under the public key pk_{i_1}. In the next steps, we can switch the witness we use to compute the proof π back to using the witness for ct, and in the last hybrid we make ct' uniformly random again. Thus, Σ is now computed using sk_{i_1}.

Compressing the Witness. The bigger challenge, however, is reducing the size of the witness for the membership proofs to linear in $\log(\ell)$. A natural approach would be to prove membership of the verification key VK_i in the ring via a Merkle-tree accumulator (as e.g. in the ROM-scheme of [14]). In this approach,

one first hashes the ring R into a succinct digest h, and can then prove membership of VK_i in the ring via a $\log(\ell)$-sized root-to-leaf path. To sign a message under a ring R, the signer first hashes R into a digest h and computes a NIWI proof π which simultaneously proves membership of his own key VK_i in R via a succinct membership witness *and* that ct encrypts a signature for VK_i. To verify such a signature, the verifier recomputes the root hash h for the ring R and verifies the proof π. While this idea seems to resolve the above issue at first glance, it raises serious issues itself. First and foremost, we will not be able to prove unforgeability as above, as membership proofs for Merkle trees only have computational soundness, but in order to prove unforgeability as above we need perfect soundness. The problem is that an adversary might also produce a proof by finding a collision in the Merkle tree instead of forging a signature. If, in fact, we could use an NIZK proof of knowledge, then this proof strategy can be implemented with routine techniques. NIZK proofs however need a setup, and we only have NIWI proofs at our disposal. Moreover, for a Merkle tree to be binding it is necessary that the hashing key is honestly generated, as unkeyed hash functions are insecure against non-uniform adversaries. Thus, it is also unclear where the hashing key for the Merkle tree should come from. Consequently, the Merkle tree approach seems fundamentally stuck in the standard model.

There is, however a loophole in the above argument. Upon closer inspection, we actually do not need the Merkle tree hash function to be collision resistant. Instead, we need a guarantee that the hash value h binds to at least one specific value in the database, which is under the control of the signer. The key ingredient we use to make the construction work is *somewhere statistically binding* (SSB) hashing [26]. An SSB hash function allows to compress a database into a digest h such that h uniquely binds to a specific database entry. More specifically, the key generator for a SSB hash function takes as an additional input an index i^* and produces a hashing key hk. When a database db is hashed into a digest h using the hashing key hk, the digest h uniquely defines db_{i^*}. In other words, any database db' with $\mathsf{db}'_{i^*} \neq \mathsf{db}_{i^*}$ hashes to a digest $\mathsf{h}' \neq \mathsf{h}$. To enable short membership proofs, we require a SSB hash function with local opening. That is, given a hashing key hk, a digest h of a database db, an index i and a value x, there is witness τ of size linear in $\log(|\mathsf{db}|)$ which demonstrates that $\mathsf{db}_i = x$. Besides the somewhere statistically binding property, we also require that the SSB hash function is index-hiding, i.e. the hashing key hk computationally hides the index i at which it is binding. Finally, as there is no trusted setup which could define the key for the SSB hash function, we must let the signer generate the hashing key hk itself. However, this again introduces an additional problem. The standard notion of SSB hashing requires that the somewhere binding property holds with overwhelming probability over the coins of the key generator, but not with probability 1. However, as we let the signer generate the hashing key, the signer may in fact choose bad random coins for which the hashing key is not binding. We address this problem by using *somewhere perfectly binding* (SPB) hashing instead of SSB hashing. In fact, many constructions of SSB hashing are already SPB, e.g. the LWE-based construction of [26] can be made SPB via stan-

dard error-truncation techniques, and the DDH- and DCR-based constructions of [32] are immediately SPB. One additional aspect we require is that generating a hashing key hk for a database db of size ℓ can be performed by a circuit of size linear in $\log(\ell)$, but this is the case for the instantiations above. Equipped with SPB hashing, we can now construct succinct membership proofs with perfect soundness as follows. The signer generates a hashing key hk binding at position i (where VK_i is the signer's verification key) and uses hk to compress R into a digest h. The membership witness shows that hk is binding at position i and that h opens to VK_i at position i. Essentially, a pair $(\mathsf{hk}, \mathsf{h})$ of SPB hashing key hk and digest h form a perfectly binding commitment to VK_i, where we can prove that $(\mathsf{hk}, \mathsf{h})$ opens to VK_i at position i using a witness of size linear in $\log(\ell)$.

Relaxing the requirements on SPB hashing. It turns out that we do not need the opening witnesses for the SPB hashing scheme to be publicly computable. Indeed, we may allow the opening witness to depend on the private coins used by the key generator as we need to prove that hk is binding at position i anyway. We therefore define a slightly weakened notion called *Somewhere Perfectly Binding Hashing with private local Opening*. As observed in [32], this notion can immediately be realized from any *private information retrieval* (PIR) scheme with fully efficient client (i.e. the clients overhead is logarithmic in the database-size). Such a PIR scheme can be immediately constructed from fully homomorphic encryption [18, 11, 19], avoiding the Merkle tree based approach of [26].

Our Scheme. Armed with these techniques, we can now provide our unlinkable ring signature scheme. Key generation is as described above. To sign a message m with a signing key sk_i, the signer computes a signature σ on m using sk_i and encrypts σ under pk_i obtaining a ciphertext ct. The ciphertext ct' is chosen uniformly random (as in the scheme above). The signer now generates two hashing keys hk and hk' which are binding at position i and computes the hash of $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ under both hk and hk', obtaining hash values h and h'. Finally, the signer computes a NIWI proof π which proves that either $(\mathsf{hk}, \mathsf{h})$ bind to a key VK_i and that ct encrypts a signature of m for VK_i *or* $(\mathsf{hk}', \mathsf{h}')$ bind to a key $\mathrm{VK}_{i'}$ and that ct' encrypts a signature of m for $\mathrm{VK}_{i'}$. The signer then outputs the signature $\Sigma = (\mathsf{ct}, \mathsf{ct}', \mathsf{hk}, \mathsf{hk}', \pi)$. To verify a signature $\Sigma = (\mathsf{ct}, \mathsf{ct}', \mathsf{hk}, \mathsf{hk}', \pi)$ for a message m and a ring $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$, the verifier first computes the hashes h and h' of R using hk and hk' respectively. Now it checks if the NIWI proof π verifies for $(\mathsf{m}, \mathsf{ct}, \mathsf{ct}', \mathsf{hk}, \mathsf{hk}', \mathsf{h}, \mathsf{h}')$, and if so it outputs 1. Unforgeabilty of this scheme is established in pretty much the same way as described above: If the proof π verifies, then by the somewhere perfectly binding property of SPB and the perfect soundness of the NIWI proof, one of the two ciphertexts $\mathsf{ct}, \mathsf{ct}'$ must in fact encrypt a valid signature. The unforgeability reduction can now recover this signature by setting up the pk_i such that it knows a secret key for each of them and can therefore recover a forge. The idea of establishing anonymity can be outlined as follows. From a high level proof perspective, SPB hashing allows us to *collapse* a ring R of ℓ verification keys into a ring of just two keys. In other words, we only care about the keys to which $(\mathsf{hk}, \mathsf{h})$ and $(\mathsf{hk}', \mathsf{h}')$ bind. With

this in mind, we can essentially implement the same proof strategy as before, pretending that our ring just consists of two keys. As before, we will transform a signature computed using a signing key sk_{i_0} into a signature computed using sk_{i_1} via a sequence of hybrids. In the first hybrid, we use the index hiding property of the SPB hash function to move the binding index of hk' from i_0 to i_1. Next, we proceed in a similar way as above, namely compute a signature σ' using sk_{i_1} and encrypt σ' under pk_{i_1} obtaining a ciphertext ct'. Indistinguishability of this hybrid from the previous hybrid can be argued via the pseudorandom ciphertexts property of PKE. In the next step, we switch the witness used to compute the NIWI proof π. That is, instead of proving that ct encrypts a valid signature under pk_{i_0}, we prove that ct' encrypts a valid signature under pk_{i_1}. Both are valid witnesses as we are proving an or-statement. Therefore, witness-indistinguishability of NIWI yields that this hybrid is indistinguishable from the last one. We can now perform the same hybrid modifications to hk and ct and finally switch the witness again. Therefore, in the last hybrid we get a signature Σ computed using sk_{i_1}. For details on this construction, refer to Section 4.

Definitions of Linkable Anonymity. The exact definition of linkable anonymity seems to vary between different authors. However, it seems that all these definitions assume that there always remain *unspent* verification keys in an anonymity set. Take for instance the definition of linkable anonymity in [29] (Definition 10 on page 13). Their definition of linkable anonymity is essentially the same as the definition of unlinkable anonymity, with the difference that the adversary is not given access to a signing oracle. We propose a simple definition for linkable anonymity similar in spirit to the blindness definition of blind signatures. The experiment is essentially identical to the anonymity experiment for unlinkable ring signatures, with the following modification:

- The adversary is not allowed to corrupt the challenge keys VK_{i_0} and VK_{i_1}
- In the challenge phase, the adversary submits two message-ring pairs $(\mathsf{m}_0, \mathsf{R}_0)$ and $(\mathsf{m}_1, \mathsf{R}_1)$ such that both R_0 and R_1 contain both VK_{i_0} and VK_{i_1}.
- The experiment flips a bit $b \leftarrow_{\$} \{0,1\}$, computes $\Sigma_0 \leftarrow \mathsf{Sign}(\mathrm{SK}_{i_b}, \mathsf{m}_0, \mathsf{R}_0)$ and $\Sigma_1 \leftarrow \mathsf{Sign}(\mathrm{SK}_{i_{1-b}}, \mathsf{m}_1, \mathsf{R}_1)$ and returns (Σ_0, Σ_1) to the adversary
- The adversary must now guess bit b.

Note that the signature Σ_0 is computed exactly as in the experiment for unlinkable anonymity, but now we additionally provide the adversary with a signature Σ_1 computed with the signing key $\mathrm{SK}_{i_{1-b}}$. Consequently, this definition immediately implies e.g. the definition of [29], but does not impose the restriction that no signatures under $\mathrm{VK}_{i_{1-b}}$ can be issued. Like the blindness definition for blind signatures, our definition naturally extends to larger challenge spaces, i.e. considering challenges of size 2 is complete. For details on this new definition refer to Section 5.

A linkable Ring Signature Scheme. We will now extend our techniques to the setting of linkable ring signatures. The underlying idea is rather basic. Every verification key VK contains a commitment com to a random tag tag. When

a signer signs a message m, he includes tag into the signature Σ and proves that com unveils to tag. This proof can naturally be included in the NIWI proof for the validity of the encrypted signature. Now, whenever a secret key SK is used to sign a message m, its corresponding tag tag is spent. Thus, we can link signatures by checking whether they have the same tag.

While this idea seems to check out at first glance, we run into trouble when trying to prove linkable anonymity. In the linkable anonymity experiment the adversary gets to see the tags of both challenge signatures. This means the reduction must be able to provide witnesses that both the commitment in VK_{i_0} and the commitment in VK_{i_1} open to the respective tags tag_{i_0} and tag_{i_1}. The fact that we need to be able to open both commitments, however, makes it apparently impossible to use the hiding property of the commitments in order to flip the challenge bit in the security proof. Once again, the situation could be resolved easily if we had NIZK proofs at our disposal, yet we can only use witness indistinguishability.

Our way out of this conundrum is based on the following observation. To achieve linkability, we do not actually need that every verification key has a unique tag. Instead, a weaker condition is sufficient. Namely, for a ring of size ℓ it should not be possible to generate $\ell+1$ valid signatures with pairwise distinct tags. We leverage this idea by allowing the commitments in the verification keys to be malformed *in a controlled way*. More specifically, instead of putting only one commitment to a tag tag in the verification key VK, we put 3 commitments to tag in VK.

As before, each signature contains two hashing keys and two hash values. Moreover, in the linkable anonymity proof we will set up things in a way such that for both challenge signatures Σ_0 and Σ_1, one of $(\mathsf{hk}, \mathsf{h})$ and $(\mathsf{hk}', \mathsf{h}')$ will point to VK_{i_0} and the other one to VK_{i_1}. Assume that a signature Σ contains a tag tag and that the SPB hash $(\mathsf{hk}, \mathsf{h})$ points to VK_i, whereas $(\mathsf{hk}', \mathsf{h}')$ points to $\mathrm{VK}_{i'}$. We will make the following consistency requirement: If $i = i'$ we will require that all three commitments in VK_i unveil to the same tag tag. However, if $i \neq i'$, then we only require that out of the six commitments in VK_i and $\mathrm{VK}_{i'}$ that

- at least two unveil to tag,
- at least two unveil to a tag $\mathsf{tag}' \neq \mathsf{tag}$,
- at most one commitment does not unveil correctly.

This relaxed binding condition now allows us to exchange the tags of VK_i and $\mathrm{VK}_{i'}$ even though we are handing out signatures which use these tags! We prove linkable anonymity via a sequence of hybrids. As above, it is instructive to think that SPB hashing collapses a ring R of ℓ keys into a ring of just two verification keys. Call these verification keys VK_0 and VK_1. In the linkable anonymity experiment, there are two signatures, Σ_0 and Σ_1 for m_0 and m_1 respectively. In the first hybrid the challenge bit of the experiment is 0, that is Σ_0 is computed using the signing key SK_0 whereas Σ_1 is computed using SK_1. In the final experiment, Σ_0 will be computed using SK_1 and Σ_1 will be computed using SK_0. The critical part of this proof is to switch the tags. Our proof strategy relies critically on the

fact that the tags tag_0 and tag_1 are identically distributed. Namely, we will not switch the tags in the signatures, but switch the tags in the verification keys. More specifically, in the first hybrid VK_{i_0} contains commitments to tag_0 and VK_{i_1} contains commitments to tag tag_1. In the last hybrid, VK_0 will commit to tag_1 and VK_1 will commit to tag_0. But since the tags are identically distributed we can now simply rename them. Therefore, this hybrid is identical to the linkable anonymity experiment with challenge bit 1. In a first step we make both signatures Σ_0 and Σ_1 use both keys VK_0 and VK_1 by modifying the binding indices in hk' appropriately for both signatures. Now, our relaxed binding condition allows us to exchange the tags between VK_0 and VK_1 one by one. That is, the relaxed binding condition allows us to *forget* the unveil information of one of the six commitments in VK_0 and VK_1. Say we forget the unveil information of the first commitment in VK_0. We can then turn this commitment into a commitment of tag_1. Next, we change the first commitment in VK_1 into a commitment of tag_0. We continue like this alternating between VK_0 and VK_1, until we have completely swapped tag_0 and tag_1. Note that in each step the relaxed binding condition holds, thus we can argue via witness indistinguishability and the hiding property of the underlying commitments. Finally, using random tags tag alone does not achieve the strongest notion of non-framemability, where the adversary is allowed to *steal* tags. Thus, we use an idea due to Dolev, Dwork and Naor [15] commonly used to achieve non-malleability[4]: We replace the tag tag by the verification key vk of a signature scheme Sig and additionally sign (m, Σ) with respect to vk. This, however, has the somewhat surprising consequence that we do not need the encrypted signatures anymore, we can rely entirely on the unforgeability of Sig! For details, refer to Section 6.

1.3 Related Work

After the initial work of Rivest, Shamir and Tauman [34], a number of works provided constructions in the random oracle model under various computational hardness assumptions [1, 8, 25]. The scheme of Dodis et al. [14] was the first to achieve sublinear size signatures in the ROM. Libert, Peters, Qian [27] constructed a scheme with logarithmic size ring signatures from DDH in the ROM. Schemes in the CRS model include [37, 9, 35, 12, 20, 22] achieving varying degrees of compactness but focusing mainly on practical efficiency. Standard model ring signatures were simultaneously proposed by Chow et al. [13] and by Bender, Katz, and Morselli [5]. Malavolta and Schröder [30] build setup free and constant size ring signatures assuming hardness of a variant of the knowledge of exponent assumption. Recently, Backes et al. [2] provided a standard model construction with signatures of size $\sqrt{\ell}$ from a new primitive called *signatures with flexible public key*. Linkable Ring signatures were introduced by Liu et al. [28] as linkable spontaneous anonymous group signatures. They propose a notion of linkability which requires that signatures created by the same signer using the same ring must be publicly linkable. In their security model, a scheme achieves

[4] e.g. in the construction of IND-CCA secure encryption schemes

a weaker, non-adaptive model of anonymity called signer-ambiguity, if given one signature under signing key SK and ring R as well as a subset of the signing keys corresponding to the keys in the ring which does not include SK, the probability of determining the actual signer as SK is at most negligibly better than guessing one of the remaining keys in the ring uniformly at random. This model is extended by Boyen and Haies [10], introducing signing epochs which allow for forward secure notions of anonymity and unforgeability. Recently, several works described linkable ring signature schemes in post-quantum setting, e.g. [38] based on the hardness of the Ring-SIS problem or [4] based on the Module-SIS and Module-LWE problems. Finally, the idea of replacing NIZK proofs with NIWI proofs in standard model constructions has gained momentum recently, e.g. in the construction of verifiable random functions (VRFs) [6, 23].

2 Preliminaries

We will denote by $y \leftarrow \mathcal{A}(x;\mathsf{r})$ the execution of algorithm $\mathcal{A}$ outputting y, on input x and random coins r. We will write $y \leftarrow \mathcal{A}(x)$ if the specific random coins used are not important. By $r \leftarrow_{\$} S$ we denote that r is chosen uniformly at random from the set S. We will use $[n]$ to denote the set $\{1,\ldots,n\}$. When defining experiments we implicitly assume the procedures take as input 1^λ as well as some additional parameters which should be clear form the context. We will use the symbol $\emptyset$ to denote an undefined value.

2.1 Signature Schemes

Definition 1. *A signature scheme* Sig *consists of three PPT algorithms* (KeyGen, $\mathsf{Sign}, \mathsf{Verify}$) *with the following syntax.*

$\mathsf{KeyGen}(1^\lambda)$**:** *Takes as input the security parameter* 1^λ *and outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$**:** *Takes as input a signing key* sk *and a message* m *and outputs a signature* σ.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma)$**:** *Takes as input a verification key* vk*, a message* m *and a signature* σ *and outputs either* 0 *or* 1.

We require the following properties of a signature scheme.

Correctness: *It holds for every security parameter* $\lambda \in \mathbb{N}$ *and every message* m *that given that* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$*,* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{m})$*, then it holds that* $\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{m}, \sigma) = 1$.

Existential Unforgeability under Chosen Message Attacks: *It holds that every PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{EUF-CMA}}(\mathcal{A})$**:** *1. The experiment generates a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ *and provides* vk *to* $\mathcal{A}$.

2. *$\mathcal{A}$ is allowed to make signing queries of the form* $(\mathtt{sign}, \mathsf{m})$*, upon which the experiment computes* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{m})$*. Further the experiment keeps a list of all signing queries.*
3. *Once $\mathcal{A}$ outputs a pair* (m^*, σ^*)*, the experiment checks if* m^* *was not queried in a signing query and if it holds that* $\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{m}^*, \sigma^*) = 1$*. If so it outputs* 1*, otherwise* 0*.*

The advantage of $\mathcal{A}$ is defined by $\mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathsf{EUF-CMA}}(\mathcal{A}) = 1]$.

2.2 Non-Interactive Commitment Schemes

Definition 2. *A commitment scheme* Com *syntactically consists of two PPT algorithms* (Commit, Verify) *with the following syntax.*

$\mathsf{Commit}(1^\lambda, \mathsf{m})$**:** *Takes as input a security parameter* 1^λ*, a message* m *and outputs a commitment* com *and unveil information* γ*.*

$\mathsf{Verify}(\mathsf{com}, \mathsf{m}, \gamma)$**:** *Takes as input a commitment* com*, a message* m *and unveil information* γ *and outputs either* 0 *or* 1*.*

We require the following properties of a signature scheme.

Correctness: *It holds for every message* m *that given* $(\mathsf{com}, \gamma) \leftarrow \mathsf{Commit}(\mathsf{m})$*, it holds that* $\mathsf{Verify}(\mathsf{com}, \mathsf{m}, \gamma) = 1$*.*

Perfect Binding: *It holds that every unbounded adversary $\mathcal{A}$ that:*

$$\Pr\left[\begin{array}{l}(\mathsf{com}, \mathsf{m}_0, \gamma_0, \mathsf{m}_1, \gamma_1) \leftarrow \mathcal{A} \ : \\ \quad \mathsf{m}_0 \neq \mathsf{m}_1 \wedge \mathsf{Verify}(\mathsf{com}, \mathsf{m}_0, \gamma_0) = \mathsf{Verify}(\mathsf{com}, \mathsf{m}_1, \gamma_1) = 1\end{array}\right] = 0$$

Computational Hiding: *We say that a commitment scheme* $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Verify})$ *is computationally hiding if for every pair of messages* $(\mathsf{m}_0, \mathsf{m}_1)$ *it holds that*

$$\mathsf{com}_0 \approx_c \mathsf{com}_1,$$

where $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{Commit}(1^\lambda, \mathsf{m}_0)$ *and* $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{Commit}(1^\lambda, \mathsf{m}_1)$*. We denote the advantage of $\mathcal{A}$ in distinguishing the commitments as* $\mathsf{Adv}_{\mathsf{Hiding}}(\mathcal{A})$*.*

Non-interactive commitment schemes can be constructed from any injective one-way function via the Goldreich-Levin hardcore bit [21].

2.3 Public Key Encryption

Definition 3. *A public key encryption scheme* PKE *consists of 3 PPT algorithms* (KeyGen, Enc, Dec) *with the following syntax.*

$\mathsf{KeyGen}(1^\lambda)$**:** *Takes as input a security parameter* 1^λ *and outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*.*

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:** *Takes as input a public key* pk *and a message* m *and outputs a ciphertext* ct

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** *Takes as input a secret key* sk *and a ciphertext* ct *and outputs a message* m *or* $\perp$

We require the following properties of a public key encryption scheme.

Perfect Correctness: *We say a public key encryption scheme* PKE *is perfectly correct, if it holds for all security parameters* $\lambda \in \mathbb{N}$ *and all messages* m *that given that* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$*,* $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{m})$*, then it holds that* $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m}$.

Pseudorandom Public Keys: *We require that public keys are computationally indistinguishable from uniform.*

Pseudorandom Ciphertexts: *We require that it holds for every message* m *that*

$$(\mathsf{pk}, u) \approx_c (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m})),$$

where pk *and* u *are chosen uniformly at random.*

We denote the advantages of $\mathcal{A}$ *in breaking pseudorandom public keys and pseudorandom public keys as* $\mathsf{Adv}_{\mathsf{IND\text{-}PK}}(\mathcal{A})$ *and* $\mathsf{Adv}_{\mathsf{IND\text{-}ENC}}(\mathcal{A})$ *respectively. Note that the pseudorandom public keys and pseudorandom ciphertext properties together immediately imply the standard notion of IND-CPA security.*

Such public key encryption schemes can be constructed e.g. from the DDH-problem [17] or the LWE-problem [33].

2.4 Somewhere Perfectly Binding Hashing

Somewhere statistically binding (SSB) hashing [26] allows a negligible fraction of the hashing-keys to be non-binding. For our constructions we actually only require something slightly weaker, a primitive we call *somewhere perfectly binding hashing with private local opening*. This notion relaxes the definition of somewhere perfectly binding hashing in that we allow the Gen algorithm to output a private key shk which the Open algorithm takes as additional input. Below we give our relaxed definition which we use throughout our paper. For completeness we recall the original definition of SSB hashing [26] in Sect. A.1. We further remark that the LWE-based construction of SBB hashing in [26] can be made somewhere perfectly binding by a noise truncation argument, and the DDH- and DCR-based schemes of [32] are immediately somewhere perfectly binding. In Sect. A.2 we recall the DDH-based SSB construction of [32] and show that it fulfils the stronger notion of *Somewhere Perfectly Binding*.

Definition 4. *A somewhere perfectly binding hash family with private local opening* SPB *is given by a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ *with the following syntax.*

$\mathsf{Gen}(1^\lambda, n, \mathsf{ind})$**:** *Takes as input a security parameter* 1^λ*, a database size* n *and an index* ind *and outputs a hashing key* hk *and a private key* shk.

$\mathsf{Hash}(\mathsf{hk}, \mathsf{db})$**:** *Takes as input a hashing key* hk *and a database* db *and outputs a digest* h.

$\mathsf{Open}(\mathsf{hk}, \mathsf{shk}, \mathsf{db}, \mathsf{ind})$**:** *Takes as input a hashing key* hk*, a private key* shk *a database* db *and an index* ind *and outputs a witness* τ*.*

$\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau)$**:** *Takes as input a hashing key* hk*, a digest* h*, an index* ind*, a value* x *and a witness* τ *and outputs either* 0 *or* 1*.*

Again, to simplify notation, we will not provide the block size of databases as an input to $\mathsf{SPB.Gen}$ *but rather assume that the block size for the specific application context is hardwired in this function. We require the following properties.*

Correctness: *We say that* $\mathsf{SPB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ *is correct, if it holds for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* db *of size* n *and all indices* $\mathsf{ind} \in [n]$ *that given that* $(\mathsf{hk}, \mathsf{shk}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind})$*,* $\mathsf{h} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}, \mathsf{db})$ *and* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}, \mathsf{shk}, \mathsf{db}, \mathsf{ind})$*, it holds that*

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

Efficiency: *The hashing keys* hk *generated by* $\mathsf{Gen}(1^\lambda, n, \mathsf{ind})$ *and the witnesses* τ *generated by* $\mathsf{Open}(\mathsf{hk}, \mathsf{shk}, \mathsf{db}, \mathsf{ind})$ *are of size* $\log(n) \cdot \mathsf{poly}(\lambda)$*. Moreover,* $\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau)$ *can be computed by a circuit of size* $\log(n) \cdot \mathsf{poly}(\lambda)$*.*

Somewhere Perfectly Binding: *It holds for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* db *of size* n*, all indices* $i \in [n]$*, all database values* x *and all witnesses* τ *that if* $\mathsf{h} = \mathsf{SPB.Hash}(\mathsf{hk}, \mathsf{db})$ *and* $\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau) = 1$*, then it holds that* $x = \mathsf{db}_{\mathsf{ind}}$*.*

Index Hiding: *Every PPT-adversary* $\mathcal{A}$ *has at most negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{I\text{-}Hiding}}(\mathcal{A})$ *:*

1. $\mathcal{A}$ *sends* $(n, \mathsf{ind}_0, \mathsf{ind}_1)$ *to the experiment.*
2. *The experiment chooses a random bit* $b \leftarrow_\$ \{0,1\}$*, computes* $(\mathsf{hk}, \mathsf{shk}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind}_b)$ *and provides* hk *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a guess* b'*. If* $b' = b$*, the experiment outputs* 1*, otherwise* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{I\text{-}Hiding}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}_{\mathsf{I\text{-}Hiding}}(\mathcal{A}) = 1] - \frac{1}{2}\right|$

Notice that this definition provides a stronger somewhere perfectly binding guarantee in that we do not have to require that hk has been generated correctly. We can immediately construct a SPB hash family SPB with private local opening from any SPB hash family SPB' with local opening via the following construction.

$\mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind})$**:**

Choose random coins $\mathsf{r} \leftarrow \{0,1\}^\lambda$, compute $\mathsf{hk} \leftarrow \mathsf{SPB}'.\mathsf{Gen}(1^\lambda, n, \mathsf{ind}; 1^\lambda; \mathsf{r})$ and output hk and $\mathsf{shk} \leftarrow \mathsf{r}$.

$\mathsf{SPB.Hash}(\mathsf{hk}, \mathsf{db})$:

Output $\mathsf{SPB}'.\mathsf{Hash}(\mathsf{hk}, \mathsf{db})$.

$\mathsf{SPB.Open}(\mathsf{hk}, \mathsf{shk} = \mathsf{r}, \mathsf{db}, \mathsf{ind})$**:**

Compute $\tau' \leftarrow \mathsf{SPB}'.\mathsf{Open}(\mathsf{hk}, \mathsf{db}, \mathsf{ind})$ and output $\tau \leftarrow (\tau', \mathsf{r})$.

$\mathsf{SPB.Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau = (\tau', \mathsf{r}))$**:**

If $\mathsf{SPB}'.\mathsf{Gen}(1^\lambda, n, \mathsf{ind}; \mathsf{r}) = \mathsf{hk}$ and $\mathsf{SPB}'.\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau') = 1$ output 1, otherwise 0.

Correctness and index-hiding of SPB follow directly from the corresponding properties of SPB', the somewhere perfectly binding property follows from the fact that $\mathsf{SPB.Verify}$ ensures explicitly that hk is perfectly binding at index ind. Consequently, also this property follows from the corresponding property of SPB'. Moreover, we can also realize a SPB hash family with private local opening from any 2-message private information retrieval scheme with fully efficient verifier and perfect correctness. This was also observed by [32]. The construction is straightforward: A hashing key hk for index i consists of the PIR receiver message, to hash a database db run the PIR sender algorithm on hk and db. The index hiding property follows by PIR receiver privacy, whereas the SPB property follows form perfect correctness. Finally, the receivers private coins serve as succinct private membership witness.

2.5 Non-Interactive Witness-Indistinguishable Proof Systems

Let $\mathcal{R}$ be an efficiently computable binary relation, where for $(x, w) \in \mathcal{R}$ we call x a statement and w a witness. Moreover, we denote by $\mathcal{L}_{\mathcal{R}}$ the language consisting of statements in $\mathcal{R}$, i.e. $\mathcal{L}_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$.

Definition 5 (Non-Interactive Proof System). *Let $\mathcal{R}$ be an efficiently computable witness relation and $\mathcal{L}_{\mathcal{R}}$ be the language accepted by $\mathcal{R}$. A non-interactive witness-indistinguishable (NIWI) proof system NIWI for $\mathcal{L}_{\mathcal{R}}$ consists of two algorithms $(\mathsf{Prove}, \mathsf{Verify})$ with the following syntax.*

$\mathsf{Prove}(1^\lambda, x, w)$**:** *Takes as input a security parameter 1^λ, a statement x and a witness w, output either a proof π or $\perp$.*

$\mathsf{Verify}(x, \pi)$**:** *Takes as input a statement x, a proof π and outputs either 0 or 1.*

We require the following properties.

Perfect Completeness: *It holds for all security parameters $\lambda \in \mathbb{N}$, all statements $x \in \mathcal{L}_{\mathcal{R}}$ and all witnesses w that if $\mathcal{R}(x, w) = 1$ and $\pi \leftarrow \mathsf{NIWI.Prove}(1^\lambda, x, w)$, then it holds that $\mathsf{NIWI.Verify}(x, \pi) = 1$.*

Perfect Soundness: *It holds for all security parameters $\lambda \in \mathbb{N}$, all statements $x \notin \mathcal{L}_{\mathcal{R}}$ and all proofs π that $\mathsf{NIWI.Verify}(x, \pi) = 0$.*

Witness-Indistinguishability: *Every PPT adversary $\mathcal{A}$ has at most negligible advantage in the the following experiment.*

$\mathsf{Exp}_{\mathsf{WI}}(\mathcal{A})$ *:*

- *$\mathcal{A}$ sends (x, w_0, w_1) with $\mathcal{R}(x, w_0) = 1$ and $\mathcal{R}(x, w_1) = 1$ to the experiment.*
- *The experiment chooses a random bit $b \leftarrow_\$ \{0, 1\}$, computes $\pi^* \leftarrow \mathsf{Prove}(1^\lambda, x, w_b)$ and provides π^* to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a guess b'. If $b' = b$ the experiment outputs 1, otherwise 0.*

The advantage of $\mathcal{A}$ is defined by $\mathsf{Adv}_{\mathsf{WI}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}_{\mathsf{WI}}(\mathcal{A}) = 1] - \frac{1}{2}\right|$.

Proof-Size: *For $\pi = \mathsf{NIWI.Prove}(1^\lambda, x, w)$ it holds that $|\pi| = |C_x| \cdot \mathsf{poly}(\lambda)$, where C_x is a verification circuit for the statement x, i.e. $(x, w) \in \mathcal{R}$ iff $C_x(w) = 1$.*

Non-interactive witness-indistinguishable proofs can be constructed from NIZK proofs and derandomization assumptions [16, 3], from bilinear pairings [24] and indistinguishablity obfuscation [7].

3 Ring-Signatures

In this section we provide the definitions related to ring signatures.

Definition 6 (Ring Signatures). *A ring signature scheme* RS *is given by a triple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *such that*

$\mathsf{KeyGen}(1^\lambda)$**:** *takes as input the security-parameter* 1^λ *and outputs a pair* $(\mathrm{VK}, \mathrm{SK})$ *of verification and signing keys.*

$\mathsf{Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$**:** *takes as input a signing key* SK*, a message* $\mathsf{m} \in \mathcal{M}_\lambda$ *and a list of verification keys* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$*, and outputs a signature* Σ*.*

$\mathsf{Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$**:** *takes as input a ring* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$*, a message* $\mathsf{m} \in \mathcal{M}_\lambda$ *and a signature* Σ*, and outputs either* 0 *or* 1*.*

Correctness: *We say that a ring signature scheme* $\mathsf{RS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is correct, if it holds for all* $\lambda \in \mathbb{N}$*, all* $\ell = \mathsf{poly}(\lambda)$*, all* $i^* \in [\ell]$ *and all messages* $\mathsf{m} \in \mathcal{M}_\lambda$ *that if for* $i \in [\ell]$ $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{RS.KeyGen}(1^\lambda)$ *and* $\Sigma \leftarrow \mathsf{RS.Sign}(\mathrm{SK}_i, \mathsf{m}, \mathsf{R})$*, where* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$*, then it holds that*

$$\Pr[\mathsf{RS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma) = 1] = 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins used by $\mathsf{RS.KeyGen}$ *and* $\mathsf{RS.Sign}$*.*

Anonymity: *We say that a ring signature scheme* $\mathsf{RS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is anonymous against full key exposure, if for every* $q = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$ *it holds that* $\mathcal{A}$ *has at most negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{RS\text{-}Anon}}(\mathcal{A})$*:*

1. *For all* $i = 1, \ldots, q$ *the experiment generates the keypairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{RS.KeyGen}(1^\lambda, \mathsf{r}_i)$ *using random coins* r_i*.*
2. *The experiment provides* $\mathrm{VK}_1, \ldots, \mathrm{VK}_q$ *and* $\mathsf{r}_1, \ldots, \mathsf{r}_q$ *to* $\mathcal{A}$*.*
3. *The adversary provides a challenge* $(\mathsf{R}, \mathsf{m}, i_0, i_1)$ *to the experiment, such that* VK_{i_0} *and* VK_{i_1} *are in the ring* R*. The experiment flips a random bit* $b \leftarrow_\$ \{0, 1\}$*, computes* $\Sigma^* \leftarrow \mathsf{RS.Sign}(\mathrm{SK}_{i_b}, \mathsf{m}, \mathsf{R})$ *and outputs* Σ^* *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* b'*. If* $b' = b$*, the experiment outputs* 1*, otherwise* 0

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{RS\text{-}Anon}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}_{\mathsf{RS\text{-}Anon}}(\mathcal{A}) = 1] - \frac{1}{2}\right|$.

Note: *We allow that the ring* R *chosen by* $\mathcal{A}$ *in step 3 may contain maliciously chosen verification keys that were not generated by the challenger.*

Unforgeability: *We say that a ring signature scheme* $\mathsf{RS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is unforgeable with respect to insider corruption, if for every* $q = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, it holds that* $\mathcal{A}$ *has at most negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{RS\text{-}Unf}}(\mathcal{A})$:

1. *For all* $i = 1, \ldots, q$ *the experiment generates the keypairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{RS.KeyGen}(1^\lambda, \mathsf{r}_i)$ *using random coins* r_i. *It sets* $\mathcal{VK} = \{\mathrm{VK}_1, \ldots, \mathrm{VK}_q\}$ *and initializes a set* $\mathcal{C} = \emptyset$.
2. *The experiment provides* $\mathrm{VK}_1, \ldots, \mathrm{VK}_q$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *is now allowed to make the following queries:*
   - $(\mathtt{sign}, i, \mathsf{m}, \mathsf{R})$ *: Upon a signing query, the experiment checks if* $\mathrm{VK}_i \in \mathsf{R}$, *and if so computes* $\Sigma \leftarrow \mathsf{RS.Sign}(\mathrm{SK}_i, \mathsf{m}, \mathsf{R})$ *and returns* Σ *to* $\mathcal{A}$. *Moreover, the experiment keeps a list of all signing queries.*
   - $(\mathtt{corrupt}, i)$ *: Upon a corruption query, the experiment adds* VK_i *to* $\mathcal{C}$ *and returns* r_i *to* $\mathcal{A}$.
4. *In the end,* $\mathcal{A}$ *outputs a tuple* $(\mathsf{R}^*, \mathsf{m}^*, \Sigma^*)$. *If it holds that* $\mathsf{R}^* \subseteq \mathcal{VK} \backslash \mathcal{C}$ *(i.e. none of the keys in* R^* *were corrupted),* $\mathcal{A}$ *never made a signing-query of the form* $(\mathtt{sign}, \cdot, \mathsf{m}^*, \mathsf{R}^*)$ *and it holds that*

$$\mathsf{RS.Verify}(\mathsf{R}^*, \mathsf{m}^*, \Sigma^*) = 1,$$

then the experiment outputs 1, *otherwise* 0.

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{RS\text{-}Unf}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathsf{RS\text{-}Unf}}(\mathcal{A}) = 1]$.

4 Construction of Ring-Signatures

In this section we will provide a construction of a ring signature scheme. Let

- $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme with pseudo-random keys and ciphertexts,
- $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme,
- $\mathsf{SPB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ be a somewhere perfectly binding hash function with private local opening and,
- $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ be a NIWI-proof system for the language $\mathcal{L}$ defined as follows. We define a witness-relation $\mathcal{R}$: If $x = (\mathsf{m}, \mathsf{ct}, \mathsf{hk}, \mathsf{h})$ and $w = (\mathrm{VK}, \mathsf{ind}, \tau, \sigma, \mathsf{r}_{\mathsf{ct}})$, where $\mathrm{VK} = (\mathsf{vk}, \mathsf{pk})$, let

$$\begin{aligned}\mathcal{R}(x, w) \Leftrightarrow\ & \mathsf{SPB.Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, \mathrm{VK}, \tau) = 1 \\ & \text{and } \mathsf{PKE.Enc}(\mathsf{pk}, \sigma; \mathsf{r}_{\mathsf{ct}}) = \mathsf{ct} \\ & \text{and } \mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{m}, \sigma) = 1\end{aligned}$$

and let $\mathcal{L}'$ be the language accepted by $\mathcal{R}$. Now, define the language $\mathcal{L}$ by

$$\mathcal{L} = \{(\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2) \mid (\mathsf{m}, \mathsf{ct}_1, \mathsf{hk}_1, \mathsf{h}_1) \in \mathcal{L}' \text{ or } (\mathsf{m}, \mathsf{ct}_2, \mathsf{hk}_2, \mathsf{h}_2) \in \mathcal{L}'\}.$$

Our ring signature scheme $\mathsf{RS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is given as follows.

$\mathsf{RS.KeyGen}(1^\lambda; \mathsf{r} = (\mathsf{r}_{\mathsf{Sig}}, \mathsf{r}_{\mathsf{pk}}))$:

- Compute $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda; \mathsf{r}_{\mathsf{Sig}})$
- Compute $\mathsf{pk} \leftarrow \mathsf{r}_{\mathsf{pk}}$
- Output $\mathrm{VK} \leftarrow (\mathsf{vk}, \mathsf{pk})$ and $\mathrm{SK} \leftarrow (\mathsf{sk}, \mathrm{VK})$

$\mathsf{RS.Sign}(\mathrm{SK} = (\mathsf{sk}, \mathrm{VK}), \mathsf{m}, \mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell))$:
- Parse $\mathrm{VK} = (\mathsf{vk}, \mathsf{pk})$
- Compute $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{m})$
- Find an index $\mathsf{ind} \in [\ell]$ such that $\mathrm{VK}_{\mathsf{ind}} = \mathrm{VK}$
- Compute $(\mathsf{hk}_1, \mathsf{shk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind})$
- Compute $(\mathsf{hk}_2, \mathsf{shk}_2) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind})$
- Compute $\mathsf{h}_1 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_1, \mathsf{R})$
- Compute $\mathsf{h}_2 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_2, \mathsf{R})$
- Compute $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1, \mathsf{shk}_1, \mathsf{R}, \mathsf{ind})$
- Compute $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \sigma; \mathsf{r}_{\mathsf{ct}})$
- Compute $\mathsf{ct}_2 \leftarrow_{\$} \{0,1\}^\lambda$
- Set $x \leftarrow (\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2)$ and $w \leftarrow (\mathrm{VK}, \mathsf{ind}, \tau, \sigma, \mathsf{r}_{\mathsf{ct}})$
- Compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$
- Output $\Sigma \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \pi)$

$\mathsf{RS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$:
- Parse $\Sigma = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \pi)$
- Compute $\mathsf{h}'_1 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_1, \mathsf{R})$
- Compute $\mathsf{h}'_2 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_2, \mathsf{R})$
- Output $\mathsf{NIWI.Verify}((\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}'_1, \mathsf{h}'_2), \pi)$

4.1 Correctness

We will first show that our scheme is correct. Assume that $\mathrm{VK} = (\mathsf{vk}, \mathsf{pk})$ and $\mathrm{SK} = (\mathsf{sk}, \mathrm{VK})$ were generated by $\mathsf{RS.KeyGen}$ and $\Sigma = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \pi)$ is the output of $\mathsf{RS.Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$, where $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$. We will show that it holds that $\mathsf{RS.Verify}(\mathsf{R}, \mathsf{m}, \sigma) = 1$. First note that since $\mathsf{SPB.Hash}$ is deterministic, it holds that $\mathsf{h}'_1 = \mathsf{h}_1$ and $\mathsf{h}'_2 = \mathsf{h}_2$. Also, it obviously holds that $\mathrm{VK} = \mathrm{VK}_{\mathsf{ind}}$ (where ind is the index of VK in R). Now, notice further that by the correctness of SPB it holds that $\mathsf{SPB.Verify}(\mathsf{hk}_1, \mathsf{h}_1, \mathsf{ind}, \mathrm{VK}_{\mathsf{ind}}, \tau) = 1$. Moreover, by the correctness of Sig it holds that $\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{m}, \sigma) = 1$. Consequently, $(\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, , \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2) \in \mathcal{L}$ and $w = (\mathrm{VK}, \mathsf{ind}, \tau, \sigma, \mathsf{r}_{\mathsf{ct}})$ is a witness for membership. Thus, by the correctness of NIWI it holds that

$$\mathsf{NIWI.Verify}((\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2), \pi) = 1$$

and consequently $\mathsf{RS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$ outputs 1.

4.2 Signature Size

For a signature $\Sigma = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \pi)$, the size of the ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ is $\mathsf{poly}(\lambda)$ and independent of the ring-size ℓ. By the efficiency property of SPB the sizes of the hashing keys $\mathsf{hk}_1, \mathsf{hk}_2$ is bounded by $\log(\ell) \cdot \mathsf{poly}(\lambda)$. Also by the efficiency property of SPB this size of the witness τ is $\log(\ell) \cdot \mathsf{poly}(\lambda)$ and the SPB-verification function Verify can be computed by a circuit of size $\log(\ell) \cdot \mathsf{poly}(\lambda)$.

Consequently, the verification circuit C_x for the language $\mathcal{L}$ and statement $x = (\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2)$ has size $\log(\ell) \cdot \mathsf{poly}(\lambda)$. By the proof-size property of the NIWI proof it holds that $|\pi| = |C_x| \cdot \mathsf{poly}(\lambda) = \log(\ell) \cdot \mathsf{poly}(\lambda)$. All together, the size of signatures Σ is $\log(\ell) \cdot \mathsf{poly}(\lambda)$.

4.3 Unforgeability

We will turn to showing that RS is unforgeable.

Theorem 1. *The ring signature scheme* RS *is unforgeable, given that* NIWI *has perfect soundness,* SPB *is somewhere perfectly binding,* PKE *is perfectly correct,* PKE *has pseudorandom public keys and* Sig *is unforgeable.*

The main idea of the proof is that since the NIWI proof has perfect soundness, it must either hold that $(\mathsf{m}, \mathsf{ct}_1, \mathsf{hk}_1, \mathsf{h}_1) \in \mathcal{L}'$ or $(\mathsf{m}, \mathsf{ct}_2, \mathsf{hk}_2, \mathsf{h}_2) \in \mathcal{L}'$. If the first statement is true, then hk_1 corresponds to an index ind_1 and $\mathcal{A}$ must have produced a forge for a key $\mathrm{VK}_{\mathsf{ind}_1}$ in R. Likewise, if the second statement is true, then $\mathcal{A}$ must have produced a forge for a key $\mathrm{VK}_{\mathsf{ind}_2}$ in R.

Proof. Let $\mathcal{A}$ be a PPT-adversary against the unforgeability experiment of RS and let further $q = \mathsf{poly}(\lambda)$ an upper bound on the number of key queries of $\mathcal{A}$. Consider the following two hybrids.

$\mathcal{H}_0$: This is the real experiment.
$\mathcal{H}_1$: The same as $\mathcal{H}_0$, except that for all $i \in [q]$ the challenger generates the public keys pk_i in VK_i by $(\mathsf{pk}_i, \hat{\mathsf{sk}}_i) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ instead of choosing pk_i uniformly at random. Moreover, the challenger stores all the secret keys $(\hat{\mathsf{sk}}_i)_{i \in [q]}$.

We will first argue that $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable given that the public keys of PKE are pseudorandom.

Claim. There exists a reduction $\mathcal{R}_1$ such that $\mathsf{Adv}_{\mathsf{IND\text{-}PK}}(\mathcal{R}_1^{\mathcal{A}}) \geq |\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]|$

The reduction $\mathcal{R}_1$ is given as follows.

Reduction $\mathcal{R}_1^{\mathcal{A}}(\mathsf{pk}^*)$
- Choose an index $i^* \leftarrow_\$ [q]$ uniformly at random.
- Simulate $\mathcal{H}_0$ with the following modifications. For all indices $i < i^*$ generate $(\mathrm{VK}_i, \mathrm{SK}_i)$ as in $\mathcal{H}_0$. For $i > i^*$ generate $(\mathrm{VK}_i, \mathrm{SK}_i)$ as in $\mathcal{H}_1$.
- Generate $(\mathrm{VK}_{i^*}, \mathrm{SK}_{i^*})$ as follows:
  - Compute $(\mathsf{vk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda; \mathsf{r}_{\mathsf{Sig}})$
  - Set $\mathrm{VK}_{i^*} \leftarrow (\mathsf{vk}_{i^*}, \mathsf{pk}^*)$ and $\mathrm{SK}_{i^*} \leftarrow (\mathsf{sk}_{i^*}, \mathrm{VK}_{i^*})$
- Output whatever the simulated experiment outputs.

Let PK_0 be the uniform distribution and PK_1 be a distribution sampled by computing $(\mathsf{pk}^*, \hat{\mathsf{sk}}^*) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and outputting pk^*. First observe that when $i^* = q - 1$ and pk^* was chosen from PK_0, then $\mathcal{R}_1^{\mathcal{A}}$ perfectly simulates $\mathcal{H}_0(\mathcal{A})$. On the other hand, if $i^* = 0$ and pk^* was chosen from PK_1, then $\mathcal{R}_1^{\mathcal{A}}$ perfectly simulates $\mathcal{H}_1(\mathcal{A})$. Moreover, observe that for $j = 1, \ldots, q-1$ it holds that

$\mathcal{R}_1^{\mathcal{A}}(PK_0)|_{i^*=j-1}$ and $\mathcal{R}_1^{\mathcal{A}}(PK_1)|_{i^*=j}$ are identically distributed. Consequently, we get that

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{IND\text{-}PK}}(\mathcal{R}_1^{\mathcal{A}}) &= |\Pr[\mathcal{R}_1^{\mathcal{A}}(PK_0)] - \Pr[\mathcal{R}_1^{\mathcal{A}}(PK_1)]| \\
&= |\sum_{j=0}^{q-1} \Pr[i^* = j] \cdot (\Pr[\mathcal{R}_1^{\mathcal{A}}(PK_0)|i^* = j] - \Pr[\mathcal{R}_1^{\mathcal{A}}(PK_1)|i^* = j])| \\
&= \frac{1}{q} \cdot |(\Pr[\mathcal{R}_1^{\mathcal{A}}(PK_0)|i^* = q-1] - \Pr[\mathcal{R}_1^{\mathcal{A}}(PK_1)|i^* = 0] \\
&\quad + \sum_{j=1}^{q-1} (\Pr[\mathcal{R}_1^{\mathcal{A}}(PK_1)|i^* = j] - \Pr[\mathcal{R}_1^{\mathcal{A}}(PK_0)|i^* = j-1]))| \\
&= \frac{1}{q} \cdot |(\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1])|.
\end{aligned}$$

Claim. There exists a reduction $\mathcal{R}_2$ such that $\mathcal{R}_2^{\mathcal{A}}$ breaks the EUF-CMA security of Sig with probability $\mathsf{Adv}_{\mathcal{H}_1}(\mathcal{A})/q$.

The reduction $\mathcal{R}_2$ is given as follows.

Reduction $\mathcal{R}_2^{\mathcal{A}}(\mathrm{VK}^*)$

- Guess an index $i^* \leftarrow_{\$} [q]$. For all $i \neq i^*$ generate VK_i and SK_i as in $\mathcal{H}_1$. Generate $\mathrm{VK}_{i^*} = \mathrm{VK}^*$ as follows. Generate $(\mathsf{pk}^*, \hat{\mathsf{sk}}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and set $\mathrm{VK}^* \leftarrow (\mathsf{vk}^*, \mathsf{pk}^*)$, where vk^* is the verification key provided by the EUF-CMA experiment. Moreover, store $\hat{\mathsf{sk}}_{i^*} = \hat{\mathsf{sk}}^*$.
- If $\mathcal{A}$ asks to corrupt VK^* abort.
- If $\mathcal{A}$ sends signature query $(\mathsf{m}, \mathrm{VK}^*, \mathsf{R})$, send m to the signing oracle of the EUF-CMA game to obtain a signature σ. Compute the signature Σ by
  - Let ind^* be the index of VK^* in R.
  - Computing $(\mathsf{hk}_1, \mathsf{shk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}^*)$
  - Computing $(\mathsf{hk}_2, \mathsf{shk}_2) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}^*)$
  - Computing $\mathsf{h}_1 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_1, R)$
  - Computing $\mathsf{h}_2 \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_2, R)$
  - Computing $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1, \mathsf{shk}_1, R, \mathsf{ind}^*)$
  - Computing $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}^*, \sigma; \mathsf{r}_{\mathsf{ct}})$
  - Computing $\mathsf{ct}_2 \leftarrow_{\$} \{0,1\}^\lambda$
  - Computing

    $$\pi \leftarrow \mathsf{NIWI.Prove}((\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, , \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2), (\mathrm{VK}^*, \mathsf{ind}^*, \tau, \sigma, \mathsf{r}_{\mathsf{ct}}))$$

  - Output $\Sigma \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \pi)$
- Once $\mathcal{A}$ outputs a forge Σ^* for $(\mathsf{m}^*, \mathsf{R}^*)$, check if it is valid, that is in the query phase $\mathcal{A}$ has not requested a signature of m^* for any key in R^*, none of the keys in R^* has been corrupted and it holds that $\mathsf{RS.Verify}(\mathsf{R}, \mathsf{m}^*, \Sigma^*) = 1$. If the forge is valid proceed.

- Parse Σ^* as $\Sigma^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \mathsf{hk}_1^*, \mathsf{hk}_2^*, \pi^*)$.
- Let $|\mathsf{R}^*| = \ell$ and let $i_1, \ldots, i_\ell$ be the indices of the keys in R^*, i.e. $\mathsf{R} = (\mathrm{VK}_{i_1}, \ldots, \mathrm{VK}_{i_\ell})$.
- For $j = 1, \ldots, \ell$:
  - Compute $\check{\sigma}_1 \leftarrow \mathsf{Dec}(\hat{\mathsf{sk}}_{i_j}, \mathsf{ct}_1^*)$ and $\check{\sigma}_2 \leftarrow \mathsf{Dec}(\hat{\mathsf{sk}}_{i_j}, \mathsf{ct}_2^*)$.
  - If $\mathsf{Sig.Verify}(\mathsf{vk}^*, \mathsf{m}^*, \check{\sigma}_1) = 1$ stop and output $\check{\sigma}_1$
  - If $\mathsf{Sig.Verify}(\mathsf{vk}^*, \mathsf{m}^*, \check{\sigma}_2) = 1$ stop and output $\check{\sigma}_2$

First note that the key-pair $(\mathsf{pk}_{i^*}, \hat{\mathsf{sk}}_{i^*})$ is correct for all messages. Notice further that, unless $\mathcal{A}$ asks to corrupt VK^*, $\mathcal{H}_1$ and the simulation of $\mathcal{R}_2$ are identically distributed from the view of $\mathcal{A}$. Observe that with probability at least $1/q$ the adversary $\mathcal{A}$ does not trigger an abort. Thus, conditioned that no abort happened, from the view of $\mathcal{A}$ the index i^* is distributed uniformly random. Assume now that $\mathcal{A}$ outputs a valid forge Σ^* for $(\mathsf{m}^*, \mathsf{R}^*)$ with $\mathsf{R}^* = (\mathrm{VK}_{i_1}, \ldots, \mathrm{VK}_{i_\ell})$. By the perfect soundness of NIWI, it holds that either $(\mathsf{m}^*, \mathsf{ct}_1^*, \mathsf{hk}_1^*, \mathsf{h}_1^*) \in \mathcal{L}'$ or $(\mathsf{m}, \mathsf{ct}_2^*, \mathsf{hk}_2^*, \mathsf{h}_2^*) \in \mathcal{L}'$. Assume w.l.o.g. that $(\mathsf{m}, \mathsf{ct}_1^*, \mathsf{hk}_1^*, \mathsf{h}_1^*) \in \mathcal{L}'$. That is, there exist $(\mathrm{VK}^\dagger, \mathsf{ind}^\dagger, \tau^\dagger, \sigma^\dagger, \mathsf{r}_{\mathsf{ct}})$ with $\mathrm{VK}^\dagger = (\mathsf{vk}^\dagger, \mathsf{pk}^\dagger)$ such that

$$\begin{aligned} &\mathsf{SPB.Verify}(\mathsf{hk}_1^*, \mathsf{h}_1^*, \mathsf{ind}^\dagger, \check{\mathrm{VK}}, \check{\tau}) = 1 \\ \text{and } &\mathsf{PKE.Enc}(\mathsf{pk}^\dagger, \sigma^\dagger; \mathsf{r}_{\mathsf{ct}}) = \mathsf{ct}_1^* \\ \text{and } &\mathsf{Sig.Verify}(\mathsf{vk}^\dagger, \mathsf{m}^*, \sigma^\dagger) = 1 \end{aligned}$$

As $\mathsf{SPB.Verify}(\mathsf{hk}_1^*, \mathsf{h}_1^*, \mathsf{ind}^\dagger, \check{\mathrm{VK}}, \check{\tau}) = 1$ and $\mathsf{h}_1^* = \mathsf{SPB.Hash}(\mathsf{hk}_1^*, \mathsf{R})$ it holds by the somewhere perfectly binding property of SPB that $\mathrm{VK}^\dagger = \mathrm{VK}_{i_{\mathsf{ind}^\dagger}}$, i.e. $\mathsf{vk}^\dagger = \mathsf{vk}^\dagger_{i_{\mathsf{ind}^\dagger}}$ and $\mathsf{pk}^\dagger = \mathsf{pk}_{i_{\mathsf{ind}^\dagger}}$. Moreover, by the above it also holds that $\mathsf{ct}_1^* = \mathsf{PKE.Enc}(\mathsf{pk}_{i_{\mathsf{ind}^\dagger}}, \sigma^\dagger; \mathsf{r}_{\mathsf{ct}})$ and $\mathsf{Sig.Verify}(\mathsf{vk}_{i_{\mathsf{ind}^\dagger}}, \mathsf{m}^*, \sigma^\dagger) = 1$.

Now observe that, as i^* is uniformly random from the view of $\mathcal{A}$, it holds that $i_{\mathsf{ind}^\dagger} = i^*$ with probability at least $1/q$. Assume therefore that $i_{\mathsf{ind}^\dagger} = i^*$. As $(\mathsf{pk}_{i^*}, \hat{\mathsf{sk}}_{i^*})$ are correct for all messages, it holds that $\check{\sigma}_1 = \mathsf{PKE.Dec}(\hat{\mathsf{sk}}_{i^*}, \mathsf{ct}_1^*) = \sigma^\dagger$. Therefore it holds that $\mathsf{Sig.Verify}(\mathsf{vk}_{i_{\mathsf{ind}^\dagger}}, \mathsf{m}^*, \check{\sigma}_1) = 1$ for the signature $\check{\sigma}_1$ decrypted by $\mathcal{R}_2^{\mathcal{A}}$, i.e. $\check{\sigma}_1$ is a valid signature of m^* under vk^*. We conclude that $\mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{R}_2^{\mathcal{A}}) \geq \frac{1}{q}|\mathsf{Adv}_{\mathcal{H}_1}(\mathcal{A}) - \nu|$.

All together, as $\mathsf{Adv}_{\mathcal{H}_1}(\mathcal{A}) \geq |\mathsf{Adv}_{\mathcal{H}_0}(\mathcal{A}) - q \cdot \mathsf{Adv}_{\mathsf{IND-PK}}(\mathcal{R}_1^{\mathcal{A}})|$ and $\mathsf{Adv}_{\mathcal{H}_0}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{RS-Unf}}(\mathcal{A})$, we can conclude that

$$\mathsf{Adv}_{\mathsf{RS-Unf}}(\mathcal{A}) \leq q \cdot \mathsf{Adv}(\mathcal{R}_1^{\mathcal{A}}) + q \cdot \mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{R}_2^{\mathcal{A}}) + \nu.$$

This concludes the proof.

On Tightness. Using a public key encryption scheme with tight multi-user security, we can improve the bound on the advantage above to

$$\mathsf{Adv}_{\mathsf{RS-Unf}}(\mathcal{A}) \leq \mathsf{Adv}(\mathcal{R}_1^{\mathcal{A}}) + q \cdot \mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{R}_2^{\mathcal{A}}) + \nu.$$

However, getting rid of the q factor for $q \cdot \mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{R}_2^{\mathcal{A}})$ seems beyond the scope of current techniques.

4.4 Anonymity

We will now turn to establishing anonymity of RS.

Theorem 2. *The ring signature scheme* RS *is anonymous, given that* SPB *is index hiding,* PKE *has pseudorandom ciphertexts and* NIWI *is computationally witness-indistinguishable.*

Our strategy is to first move the index of hk_2 from i_0 to i_1 and argue indistinguishability via the index-hiding property of SPB. Next we switch ct_2 to an encryption of a signature σ' of m for the verification key VK_{i_1}. This modification will not be detected due to the pseudorandom ciphertexts prooperty of the PKE. Now, we can switch the NIWI witness to a witness for $(\mathsf{m}, \mathsf{ct}_2, \mathsf{hk}_2, \mathsf{h}_2) \in \mathcal{L}'$. Next, we perform the first two changes above for hk_1 and ct_1, switch the witness back to the witness for $(\mathsf{m}, \mathsf{ct}_1, \mathsf{hk}_1, \mathsf{h}_1) \in \mathcal{L}'$, and finally replace ct_2 with a random string. The signature in the last experiment is now a real signature of m under VK_{i_1}.

Proof. Let $\mathcal{A}$ be a PPT-adversary against the anonymity of RS. Assume that $\mathcal{A}$ makes at most $q = \mathsf{poly}(\lambda)$ key queries. Let in the following ind_0 be the index of VK_{i_0} in R and ind_1 be the index of VK_{i_1} ind R, where $(i_0, i_1, \mathsf{m}^*, \mathsf{R})$ is the challenge query of $\mathcal{A}$. Consider the following hybrids:

$\mathcal{H}_0$: This is the real experiment with challenge-bit $b^* = 0$.

$\mathcal{H}_1$: Same as $\mathcal{H}_0$, except that in Σ^* we compute hk_2^* by $(\mathsf{hk}_2^*, \mathsf{shk}_2^*) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}_1)$ instead of computing $(\mathsf{hk}_2^*, \mathsf{shk}_2^*) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}_0)$. Moreover, also compute $\tau' \leftarrow \tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_2, \mathsf{shk}_2, \mathsf{R}, \mathsf{ind}_1)$.

$\mathcal{H}_2$: Same as $\mathcal{H}_1$, except that we compute ct_2^* by
- $\sigma' \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{i_1}, \mathsf{m}^*)$
- $\mathsf{ct}_2^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i_1}, \sigma'; \mathsf{r}_{\mathsf{ct}_2})$

instead of $\mathsf{ct}_2^* \leftarrow_\$ \{0,1\}^\lambda$.

$\mathcal{H}_3$: The same as $\mathcal{H}_2$, except that we use the witness $w' \leftarrow (\mathrm{VK}_{i_1}, \mathsf{ind}_1, \tau', \sigma', \mathsf{r}_{\mathsf{ct}_2})$ instead of $w \leftarrow (\mathrm{VK}_{i_0}, \mathsf{ind}_0, \tau, \sigma, \mathsf{r}_{\mathsf{ct}_1})$ to compute π, i.e. we compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w')$.

$\mathcal{H}_4$: The same as $\mathcal{H}_3$, except that we compute ct_1^* by $\mathsf{ct}_1^* \leftarrow_\$ \{0,1\}^\lambda$.

$\mathcal{H}_5$: The same as $\mathcal{H}_4$, except that we compute hk_1^* by $(\mathsf{hk}_1^*, \mathsf{shk}_1^*) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}_1)$ instead of $(\mathsf{hk}_1^*, \mathsf{shk}_1^*) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}_0)$. Moreover, also compute τ by $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1, \mathsf{shk}_1, \mathsf{R}, \mathsf{ind}_1)$.

$\mathcal{H}_6$: The same as $\mathcal{H}_5$, except that we compute ct_1^* by
- $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{i_1}, \mathsf{m}^*)$
- $\mathsf{ct}_1^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i_1}, \sigma; \mathsf{r}_{\mathsf{ct}_1})$

instead of $\mathsf{ct}_1^* \leftarrow_\$ \{0,1\}^\lambda$.

$\mathcal{H}_7$: The same as $\mathcal{H}_6$, except that we use the witness $w'' \leftarrow (\mathrm{VK}_{i_1}, \mathsf{ind}_1, \tau, \sigma, \mathsf{r}_{\mathsf{ct}_1})$ instead of $w' \leftarrow (\mathrm{VK}_{i_1}, \mathsf{ind}_1, \tau', \sigma', \mathsf{r}_{\mathsf{ct}_2})$ to compute π, i.e. we compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w'')$.

$\mathcal{H}_8$: The same as $\mathcal{H}_7$ except that we compute ct_2^* by $\mathsf{ct}_2^* \leftarrow_\$ \{0,1\}^\lambda$. This is identical to the real experiment with $b^* = 1$.

We will show indistinguishability of the hybrids via a sequence of claims.

Claim. $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable, given that SPB is index hiding. More specifically, there exists a reduction $\mathcal{R}_1$ such that

$$\mathsf{Adv}_{\mathsf{I\text{-}Hiding}}(\mathcal{R}_1^{\mathcal{A}}) = |\Pr[\mathcal{H}_1(\mathcal{A}) = 1] - \Pr[\mathcal{H}_0(\mathcal{A}) = 1]|.$$

We will provide an informal description of $\mathcal{R}_1$. $\mathcal{R}_1$ simulates $\mathcal{H}_0$ faithfully, until $\mathcal{A}$ announces the challenge query $(i_0, i_1, \mathsf{m}^*, \mathsf{R})$. $\mathcal{R}_1$ now provides $(i_0, i_1, |\mathsf{R}|)$ to the index-hiding experiment and receives a hashing key hk^*. $\mathcal{R}_1$ continues the simulation of $\mathcal{H}_0$ faithfully, except that in the challenge ciphertext it sets $\mathsf{hk}_2 \leftarrow \mathsf{hk}^*$. In the end, $\mathcal{R}_1$ outputs whatever the simulated $\mathcal{H}_0$ outputs.

Clearly, if the challenge bit of the index hiding experiment is 0 then $\mathcal{R}_1^{\mathcal{A}}$ simulates $\mathcal{H}_0$ perfectly. On the other hand, if the challenge bit of the index hiding experiment is 1 then $\mathcal{R}_1^{\mathcal{A}}$ simulates $\mathcal{H}_1$ perfectly. The claim follows.

Claim. We claim that $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable, given that the ciphertexts of PKE are pseudorandom. More specifically, there exists a reduction $\mathcal{R}_2$ against the pseudorandomness of ciphertexts of PKE such that

$$q \cdot \mathsf{Adv}_{\mathsf{IND\text{-}ENC}}(\mathcal{R}_2^{\mathcal{A}}) \geq |\Pr[\mathcal{H}_2(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]|.$$

The reduction $\mathcal{R}_2$ receives as input a public key pk^*. $\mathcal{R}_2$ simulates $\mathcal{H}_1$ faithfully, except for the following. Before the simulation starts, $\mathcal{R}_2$ guesses an index i_1^* and sets $\mathrm{VK}_{i_1^*} \leftarrow (\mathsf{vk}_{i_1^*}, \mathsf{pk}^*)$, where $\mathsf{vk}_{i_1^*}$ is generated as in $\mathcal{H}_1$ and pk^* is the input of $\mathcal{R}_2$. $\mathcal{R}_2$ continues the simulation of $\mathcal{H}_1$ until $\mathcal{A}$ announces $(i_0, i_1, \mathsf{m}^*, \mathsf{R})$. If it holds $i_1 \neq i_1^*$, $\mathcal{R}_2$ outputs $\perp$. Otherwise, $\mathcal{R}_2$ computes $\sigma' \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{i_1}, \mathsf{m}^*)$ and sends σ' to the experiment and receives a ciphertext ct^*. In the computation of the challenge signature it sets $\mathsf{ct}_2^* \leftarrow \mathsf{ct}^*$. $\mathcal{R}_2$ now continues the simulation and outputs whatever the simulated $\mathcal{H}_1$ outputs.

Clearly, if the challenge bit of the pseudorandom ciphertexts experiment is 0, then from the view of $\mathcal{A}$ the simulation of $\mathcal{R}_2$ is identically distributed to $\mathcal{H}_0$. On the other hand, if the challenge bit of the pseudorandom ciphertexts experiment is 1, then the view of $\mathcal{A}$ is identically distributed as in $\mathcal{H}_2$. Finally, the guess of i_1^* of $\mathcal{R}_2$ is correct with probability at least $\frac{1}{q}$, thus with probability $\frac{1}{q}$ no abort happens and therefore

$$\mathsf{Adv}_{\mathsf{IND\text{-}ENC}}(\mathcal{R}_2^{\mathcal{A}}) \geq \frac{1}{q} \cdot |\Pr[\mathcal{H}_2(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]|.$$

Claim. We claim that $\mathcal{H}_2$ and $\mathcal{H}_3$ are computationally indistinguishable, given that NIWI is computationally witness-indistinguishable. More specifically, there exists a reduction $\mathcal{R}_3$ against the witness indistinguishability of NIWI such that

$$\mathsf{Adv}_{\mathsf{WI}}(\mathcal{R}_3) = |\Pr[\mathcal{H}_3(\mathcal{A}) = 1] - \Pr[\mathcal{H}_2(\mathcal{A}) = 1]|.$$

The reduction $\mathcal{R}_3$ simulates $\mathcal{H}_2$ faithfully, until the challenge signature is computed. Instead of computing the proof π itself, $\mathcal{R}_3$ sends the statement

$x = (\mathsf{m}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{hk}_1, \mathsf{hk}_2, \mathsf{h}_1, \mathsf{h}_2)$ and the witnesses $w_0 \leftarrow (\mathrm{VK}_{i_0}, \mathsf{ind}_0, \tau, \sigma, \mathsf{r}_{\mathsf{ct}_1})$ and $w_1 \leftarrow (\mathrm{VK}_{i_1}, \mathsf{ind}_1, \tau', \sigma', \mathsf{r}_{\mathsf{ct}_2})$ to the witness-indistinguishability experiment. The experiment returns a proof π^*, and $\mathcal{R}_3$ uses the proof π^* in the challenge signature. $\mathcal{R}_3$ continues the simulation of $\mathcal{H}_2$ faithfully and outputs whatever the simulated $\mathcal{H}_2$ outputs.

Clearly, if the challenge-bit of the witness-indistinguishability experiment is 0, then $\mathcal{R}_3^{\mathcal{A}}$ simulates $\mathcal{H}_2(\mathcal{A})$ perfectly. On the other hand, if the challenge bit is 1 then $\mathcal{R}_3^{\mathcal{A}}$ simulates $\mathcal{H}_3(\mathcal{A})$ perfectly. The claim follows.

Claim. We claim that $\mathcal{H}_3$ and $\mathcal{H}_4$ are computationally indistinguishable, given that the ciphertexts of PKE are pseudorandom. More specifically, there exists a reduction $\mathcal{R}_4$ against the pseudorandomness of ciphertexts of PKE such that

$$q \cdot \mathsf{Adv}_{\mathsf{IND\text{-}PK}}(\mathcal{R}_4^{\mathcal{A}}) \geq |\Pr[\mathcal{H}_4(\mathcal{A}) = 1] - \Pr[\mathcal{H}_3(\mathcal{A}) = 1]|.$$

The proof of claim follows analogously to the proof of Claim 4.4, except that we perform the change on ct_1

Claim. $\mathcal{H}_4$ and $\mathcal{H}_5$ are computationally indistinguishable, given that SPB is index hiding. More specifically, there exists a reduction $\mathcal{R}_5$ such that

$$\mathsf{Adv}_{\mathsf{I\text{-}Hiding}}(\mathcal{R}_5^{\mathcal{A}}) = |\Pr[\mathcal{H}_4(\mathcal{A}) = 1] - \Pr[\mathcal{H}_5(\mathcal{A}) = 1]|.$$

The proof follows analogously to the proof of Claim 4.4.

Claim. We claim that $\mathcal{H}_6$ and $\mathcal{H}_5$ are computationally indistinguishable, given that the ciphertexts of PKE are pseudorandom. More specifically, there exists a reduction $\mathcal{R}_6$ against the pseudorandomness of ciphertexts of PKE such that

$$q \cdot \mathsf{Adv}_{\mathsf{IND\text{-}PK}}(\mathcal{R}_6^{\mathcal{A}}) \geq |\Pr[\mathcal{H}_6(\mathcal{A}) = 1] - \Pr[\mathcal{H}_5(\mathcal{A}) = 1]|.$$

The proof follows analogously to the proof of Claim 4.4.

Claim. We claim that $\mathcal{H}_7$ and $\mathcal{H}_6$ are computationally indistinguishable, given that NIWI is computationally witness-indistinguishable. More specifically, there exists a reduction $\mathcal{R}_7$ against the witness indistinguishability of NIWI such that

$$\mathsf{Adv}_{\mathsf{WI}}(\mathcal{R}_7) = |\Pr[\mathcal{H}_7(\mathcal{A}) = 1] - \Pr[\mathcal{H}_6(\mathcal{A}) = 1]|.$$

The proof follows analogously to the proof of Claim 4.4, except that we perform the change on ct_1.

Claim. We claim that $\mathcal{H}_8$ and $\mathcal{H}_7$ are computationally indistinguishable, given that the ciphertexts of PKE are pseudorandom. More specifically, there exists a reduction $\mathcal{R}_8$ against the pseudorandomness of ciphertexts of PKE such that

$$q \cdot \mathsf{Adv}_{\mathsf{IND\text{-}ENC}}(\mathcal{R}_8^{\mathcal{A}}) \geq |\Pr[\mathcal{H}_8(\mathcal{A}) = 1] - \Pr[\mathcal{H}_7(\mathcal{A}) = 1]|.$$

The proof follows analogously to the proof of Claim 4.4.

5 Linkable Ring-Signatures

In this section we introduce our new model for linkable ring signatures.

Definition 7 (Linkable Ring Signatures.). *Syntactically, a ring signature scheme* LRS *is given by PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ *such that*

$\mathsf{KeyGen}(1^\lambda)$**:** *takes as input the security-parameter* 1^λ *and outputs a pair* $(\mathrm{VK}, \mathrm{SK})$ *of verification and signing keys.*

$\mathsf{Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$**:** *takes as input a signing key* SK*, a message* $\mathsf{m} \in \mathcal{M}_\lambda$ *and a list of verification keys* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_q)$*, and outputs a signature* Σ*.*

$\mathsf{Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$**:** *takes as input a ring* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_q)$*, a message* $\mathsf{m} \in \mathcal{M}$ *and a signature* Σ*, and outputs either* 0 *or* 1*.*

$\mathsf{Link}(\Sigma_1, \Sigma_2, m_1, m_2)$**:** *takes as input two signatures and two messages and outputs either* 0 *or* 1*.*

We say that a linkable ring signature scheme $\mathsf{LRS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ *is correct, if it holds for all* $\lambda \in \mathbb{N}$*, all* $q = \mathsf{poly}(\lambda)$*, all* $i^* \in [\ell]$ *and all messages* $\mathsf{m} \in \mathcal{M}_\lambda$ *that, if* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{LRS.KeyGen}(1^\lambda)$ *and* $\Sigma \leftarrow \mathsf{LRS.Sign}(\mathrm{SK}_i, \mathsf{m}, \mathsf{R})$*, where* $i \in [q]$ *and* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_q)$*, then*

$$\Pr[\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}, \sigma) = 1] = 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins used by $\mathsf{LRS.KeyGen}$ *and* $\mathsf{LRS.Sign}$*.*

We will now define security properties of linkable ring signatures and begin with the core property called linkability. Informally, we may think of it as the requirement that any two or more uses of a secret key can be publicly linked. We model this property by letting an adversary output q verification keys and signatures, where none of the signatures links with each other. In order to break linkability the adversary has to output one additional signature which does not link with any of the former signatures. Note that producing q signatures which do not link is easy. The adversary only has to use the q different secret keys. But producing the one additional signature without an additional verification key, is required to be infeasible.

Definition 8 (Linkability). *We say that a linkable ring signature scheme* $\mathsf{LRS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ *has linkability property, if for every* $q = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, it holds that* $\mathcal{A}$ *has negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{LRS\text{-}Link}}(\mathcal{A})$*:*

1. $\mathcal{A}$ *outputs a set of tuples* $(\mathrm{VK}_i, \Sigma_i, m_i, \mathsf{R}_i)$ *for* $i = 1, \ldots, q$ *and another tuple* $(\sigma^*, m^*, \mathsf{R}^*)$*. Denote as* $\mathcal{VK}$ *the set of cardinality* q *such that* $\mathrm{VK}_i \in \mathcal{VK}$ *for* $i = 1, \ldots, q$*.*
2. *The experiment outputs* 1 *if the following conditions hold:*
   - *For all* $i \in [q]$ *we have* $\mathsf{R}_i \subseteq \mathcal{VK}$ *and* $\mathsf{R}^* \subseteq \mathcal{VK}$

- *For all* $i \in [q]$ *we have* $\mathsf{LRS.Verify}(\mathsf{R}_i, \mathsf{m}_i, \Sigma_i) = 1$ *and* $\mathsf{LRS.Verify}(\mathsf{R}^*, \mathsf{m}^*, \Sigma^*) = 1$
- *For all* $i, j \in [q]$ *such that* $i \neq j$*, we have* $\mathsf{LRS.Link}(\Sigma_i, \Sigma_j) = 0$ *and* $\mathsf{LRS.Link}(\Sigma_i, \Sigma^*) = 0$

Otherwise, the experiment returns 0.

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{LRS\text{-}Link}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathsf{LRS\text{-}Link}}(\mathcal{A}) = 1]$.

We now turn to anonymity. Since, in linkable ring signatures, there is a public link function, it is easy to tell whether multiple signatures were produced by the same signer or not. However, it should still be infeasible to tell which exact user from a ring produced the signature. We argue that, in contrast to the state-of-the-art definitions, in our definition anonymity is not lost at the moment an adversary obtains the first signature of a user. In reality, even when an adversary obtains multiple signature from the same member, identity of the signer should still be unknown, i.e. it should be infeasible to associate the signatures with a verification key. We model this by letting the adversary choose two users, which need to be always in the same rings, and imposing a permutation on the secret keys. If an adversary would be able to associate a signature of one of this users with its verification key, then the adversary would also be able to guess the permutation.

Definition 9 (Linkable Anonymity.). *We say that a linkable ring signature* $\mathsf{LRS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ *is linkably anonymous, if for every* $q = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, it holds that* $\mathcal{A}$ *has negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{LRS\text{-}Anon}}(\mathcal{A})$*:*

1. *For all* $i = 1, \ldots, q$ *the experiment generates* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{LRS.KeyGen}(1^\lambda, \mathsf{r}_i)$ *using random coins* r_i *and samples* $b \in \{0, 1\}$ *uniformly at random.*
2. *The experiment provides* $\mathcal{VK} = \{\mathrm{VK}_1, \ldots, \mathrm{VK}_q\}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a set of verification keys* $\mathcal{U} \subset \mathcal{VK}$ *and two challenge verification keys* $\mathrm{VK}_0^*, \mathrm{VK}_1^* \in \mathcal{VK} \setminus \mathcal{U}$*. We denote the secret keys corresponding to* $\mathrm{VK}_0^*, \mathrm{VK}_1^*$ *as* $\mathrm{SK}_0^*, \mathrm{SK}_1^*$ *respectively. The experiment returns* r_i *for all* $\mathrm{VK}_i \in \mathcal{U}$*.*
4. *The adversary queries for signatures on input a ring* R *and a verification key* $\mathrm{VK} \in \mathcal{VK} \setminus \mathcal{U}$ *such that* $\mathrm{VK} \in \mathsf{R}$*.*
   - *If* VK_0^* *or* $\mathrm{VK}_1^* \in \mathsf{R}$ *but* $\{\mathrm{VK}_0^*, \mathrm{VK}_1^*\} \not\subseteq \mathsf{R}$*, then the experiment returns an uniformly random bit and aborts.*
   - *If* $\mathrm{VK} \notin \{\mathrm{VK}_0^*, \mathrm{VK}_1^*\}$*, then the experiment outputs* $\Sigma^* \leftarrow \mathsf{LRS.Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$ *where* SK *corresponds to the queried* VK*.*
   - *If* $\mathrm{VK} = \mathrm{VK}_0^*$ *the experiment outputs* $\Sigma^* \leftarrow \mathsf{LRS.Sign}(\mathrm{SK}_b^*, \mathsf{m}, \mathsf{R})$*.*
   - *If* $\mathrm{VK} = \mathrm{VK}_1^*$ *the experiment outputs* $\Sigma^* \leftarrow \mathsf{LRS.Sign}(\mathrm{SK}_{1-b}^*, \mathsf{m}, \mathsf{R})$*.*
5. $\mathcal{A}$ *submits* $\hat{b} \in \{0, 1\}$ *and the experiment outputs* 1 *if* $\hat{b} = b$*, otherwise it outputs* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{LRS\text{-}Anon}}(\mathcal{A}) = |\Pr[\mathsf{Exp}_{\mathsf{LRS\text{-}Anon}}(\mathcal{A}, q, \lambda) = 1] - 1/2|$.

Finally, we require that a linkable ring signature is non-frameable. This property guarantees that it is infeasible for an adversary to forge a signature which would link with an honest users' signature, even when the adversary saw a number of his signatures in the past.

Definition 10 (Non-Frameability). *We say that a linkable ring signature* $\mathsf{LRS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ *is non-frameable, if for every* $q = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, it holds that* $\mathcal{A}$ *has negligible advantage in the following experiment.*

$\mathsf{Exp}_{\mathsf{LRS\text{-}Frame}}(\mathcal{A})$*:*

1. *For all* $i = 1, \ldots, q$ *the experiment generates* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{LRS.KeyGen}(1^\lambda, \mathsf{r}_i)$ *using uniformly random coins* r_i*. The experiment sets* $\mathcal{VK} = \{\mathrm{VK}_1, \ldots, \mathrm{VK}_q\}$ *and initializes a set* $\mathcal{C} = \emptyset$*.*
2. *The experiment provides* $\mathrm{VK}_1, \ldots, \mathrm{VK}_q$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *is now allowed to make the following queries:*
   - $(\mathtt{sign}, \mathrm{VK}_i, \mathsf{m}, \mathsf{R})$ *: Upon a signing query, the experiment checks if* $\mathrm{VK}_i \in \mathsf{R}$*, and if so computes* $\Sigma \leftarrow \mathsf{LRS.Sign}(\mathrm{SK}_i, \mathsf{m}, \mathsf{R})$ *and returns* Σ *to* $\mathcal{A}$*. Note that we don't require* $\mathsf{R} \subseteq \mathcal{VK}$*, so the ring* R *may contain verification keys generated by* $\mathcal{A}$*.*
   - $(\mathtt{corrupt}, \mathrm{VK}_i)$ *: Upon a corruption query, the experiment adds* VK_i *to* $\mathcal{C}$ *and returns* r_i *to* $\mathcal{A}$*.*
4. *In the end of Phase-1,* $\mathcal{A}$ *outputs* $(\mathsf{R}^*, \mathsf{m}^*, \Sigma^*)$*.*
5. *The experiment now provides all random coins* r_i *for all* $i = 1, \ldots, q$ *used to generate the keys to the adversary* $\mathcal{A}$*.*
6. *The adversary* $\mathcal{A}$ *outputs* $(\mathsf{R}^\dagger, \mathsf{m}^\dagger, \Sigma^\dagger)$ *and the experiment returns* 1 *if the following conditions hold:*
   - $\mathsf{LRS.Verify}(\mathsf{R}^*, \mathsf{m}^*, \Sigma^*) = 1$ *and* $\mathsf{LRS.Verify}(\mathsf{R}^\dagger, \mathsf{m}^\dagger, \Sigma^\dagger) = 1$*,*
   - $\mathsf{R}^* \subseteq \mathcal{VK}$ *and for all* $\mathrm{VK}_i \in \mathsf{R}^*$ *we have* $\mathrm{VK}_i \notin \mathcal{C}$*, i.e. all verification keys in* R^* *are from honest users,*
   - $\mathcal{A}$ *didn't obtain* Σ^* *from the signing oracle,*
   - $\mathsf{Link}(\Sigma^*, \Sigma^\dagger) = 1$*.*

   Otherwise the experiment returns 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}_{\mathsf{LRS\text{-}Frame}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathsf{LRS\text{-}Frame}}(\mathcal{A}) = 1]$.

Remark 1 (Unforgeability). Beside the properties defined above, we also require the standard unforgeability property from ring signatures to hold for linkable ring signatures.

6 Construction of Linkable Ring Signatures

We will now provide a construction of linkable ring signatures from the following primitives. Let

- $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Verify})$ be a non-interactive commitment scheme.
- $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme.

– $\mathsf{SPB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ be a somewhere perfectly binding hash function with private local opening.

Before we define the NIWI-proof system NIWI, we will define an algorithm $\mathsf{JointVerify}$. The algorithm takes as input two commitment triples $\mathrm{VK} = (\mathsf{com}_j)_{j\in[3]}$ and $\mathrm{VK}' = (\mathsf{com}'_j)_{j\in[3]}$, two inputs vk and vk' as well as two unveil triples $\boldsymbol{\gamma} = (\gamma_j)_{j\in[3]}, \boldsymbol{\gamma}' = (\gamma'_j)_{j\in[3]}$. The algorithm checks that one of the commitments $\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3, \mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3$ at least two open to vk and at least two open to vk' and at least 5 open to either vk or vk'. The last condition can be rephrased as at most one of the 6 commitments does not verify and all the others open to either vk or vk'. As the name suggests, the algorithm verifies if the triples VK and VK′ jointly commit to the values vk and vk', but we allow some leeway which of the 6 commitments actually commit to which value.

$\mathsf{JointVerify}(\mathrm{VK}, \mathrm{VK}', \mathsf{vk}, \mathsf{vk}', \boldsymbol{\gamma}, \boldsymbol{\gamma}')$:
- Parse $\mathrm{VK} = (\mathsf{com}_j)_{j\in[3]}$ and $\mathrm{VK}' = (\mathsf{com}'_j)_{j\in[3]}$
- Parse $\boldsymbol{\gamma} = (\gamma_j)_{j\in[3]}$, $\boldsymbol{\gamma}' = (\gamma'_j)_{j\in[3]}$
- Compute $s \leftarrow \sum_{j=1}^{3}(\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}, \gamma_j) + \mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}, \gamma'_j))$
- Compute $s' \leftarrow \sum_{j=1}^{3}(\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}', \gamma_j) + \mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}', \gamma'_j))$
- If it holds that $s \geq 2$ and $s' \geq 2$ and $s + s' \geq 5$ output 1, otherwise 0.

We remark that the expression $\mathsf{JointVerify}(\mathrm{VK}, \mathrm{VK}', \mathsf{vk}, \mathsf{vk}', \boldsymbol{\gamma}, \boldsymbol{\gamma}') = 1$ can be unrolled into a short (constant size) sequence of conjunctions and disjunctions over expressions of the form $\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}, \gamma_j) = 1$, $\mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}, \gamma'_j) = 1$, $\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}', \gamma_j) = 1$ and $\mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}', \gamma'_j) = 1$ for $j = 1, 2, 3$[5].

– $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ be a NIWI-proof system for the language $\mathcal{L}$ and with witness-relation $\mathcal{R}$ defined as follows. For $x = (\mathsf{vk}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]})$ and $w = ((\mathsf{ind}^{(i)}, \mathrm{VK}^{(i)}, \tau^{(i)}, \boldsymbol{\gamma}^{(i)})_{i\in[3]}, \mathsf{vk}')$, where $\mathrm{VK}^{(i)} = (\mathsf{com}_1^{(i)}, \mathsf{com}_2^{(i)}, \mathsf{com}_3^{(i)})$ and $\boldsymbol{\gamma}^{(i)} = (\gamma_1^{(i)}, \gamma_2^{(i)}, \gamma_3^{(i)})$ for $i = 1, \ldots, 3$, let

$$\begin{aligned}\mathcal{R}(x, w) \Leftrightarrow\ & \mathsf{SPB.Verify}(\mathsf{hk}^{(1)}, \mathsf{h}^{(1)}, \mathsf{ind}^{(1)}, \mathrm{VK}^{(1)}, \tau^{(1)}) = 1 \\ & \text{and } \forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_j^{(1)}, \mathsf{vk}, \gamma_j^{(1)}) = 1 \\ & \text{or} \\ & \mathsf{ind}^{(2)} \neq \mathsf{ind}^{(3)} \\ & \text{and } \forall i \in \{2, 3\} : \mathsf{SPB.Verify}(\mathsf{hk}^{(i)}, \mathsf{h}^{(i)}, \mathsf{ind}^{(i)}, \mathrm{VK}^{(i)}, \tau^{(i)}) = 1 \\ & \text{and } \mathsf{JointVerify}(\mathrm{VK}^{(2)}, \mathrm{VK}^{(3)}, \mathsf{vk}, \mathsf{vk}', \boldsymbol{\gamma}^{(2)}, \boldsymbol{\gamma}^{(3)}) = 1.\end{aligned}$$

Let $\mathcal{L}$ be the language accepted by $\mathcal{R}$.

Our linkable ring signature scheme $\mathsf{LRS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is given as follows.

[5] The expression can be unrolled into a disjunction of $6 \cdot \left(\binom{5}{2} + \binom{5}{3}\right) = 480$ clauses, where each clause is a conjunction of 5 $\mathsf{Com.Verify}$ statements

$\mathsf{LRS.KeyGen}(1^\lambda)$:
- Compute $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
- For $i = 1, 2, 3$ compute $(\mathsf{com}_j, \gamma_j) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk})$
- Set $\boldsymbol{\gamma} \leftarrow (\gamma_j)_{j\in[3]}$
- Output $\mathrm{VK} \leftarrow (\mathsf{com}_j)_{j\in[3]}$ and $\mathrm{SK} \leftarrow (\mathsf{sk}, \mathrm{VK}, \mathsf{vk}, \boldsymbol{\gamma})$

$\mathsf{LRS.Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell))$:
- Parse $\mathrm{SK} = (\mathsf{sk}, \mathrm{VK}, \mathsf{vk}, \boldsymbol{\gamma})$
- Parse $\mathrm{VK} = (\mathsf{com}_j)_{j\in[3]}$
- Find an index $\mathsf{ind} \in [\ell]$ such that $\mathrm{VK}_{\mathsf{ind}} = \mathrm{VK}$
- For $i = 1, 2, 3$ compute $(\mathsf{hk}^{(i)}, \mathsf{shk}^{(i)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind})$ and $\mathsf{h}^{(i)} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}^{(i)}, \mathsf{R})$
- Compute $\tau^{(1)} \leftarrow \mathsf{SPB.Open}(\mathsf{hk}^{(1)}, \mathsf{shk}^{(1)}, \mathsf{R}, \mathsf{ind})$
- Set $x \leftarrow (\mathsf{vk}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]})$
- Set $w \leftarrow ((\mathsf{ind}, \mathrm{VK}, \tau^{(1)}, \boldsymbol{\gamma}), \emptyset, \emptyset, \emptyset)$
- Compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$
- Compute $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, (\mathsf{m}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}, \pi))$
- Output $\Sigma \leftarrow (\mathsf{vk}, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi, \sigma)$

$\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$:
- Parse $\Sigma = (\mathsf{vk}, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi, \sigma)$
- For $i \in [3]$ compute $\tilde{\mathsf{h}}^{(i)} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}^{(i)}, \mathsf{R})$
- Set $x \leftarrow (\mathsf{vk}, (\mathsf{hk}^{(i)}, \tilde{\mathsf{h}}^{(i)})_{i\in[3]})$
- Check if $\mathsf{NIWI.Verify}(x, \pi) = 1$, if not output 0
- Check if $\mathsf{Sig.Verify}(\mathsf{vk}, (\mathsf{m}, (\mathsf{hk}^{(i)}, \tilde{\mathsf{h}}^{(i)})_{i\in[3]}, \pi), \sigma) = 1$, if not output 0
- Output 1

$\mathsf{LRS.Link}(\Sigma_1, \Sigma_2)$:
- Parse $\Sigma_1 \leftarrow (\mathsf{vk}_1, (\mathsf{hk}_1^{(i)})_{i\in[3]}, \pi_1, \sigma_1)$
- Parse $\Sigma_2 \leftarrow (\mathsf{vk}_2, (\mathsf{hk}_2^{(i)})_{i\in[3]}, \pi_2, \sigma_2)$
- If $\mathsf{vk}_1 = \mathsf{vk}_2$ output 1, otherwise 0

6.1 Correctness

Again, we will first show correctness of our scheme. Assume that $\mathrm{VK} = (\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ and $\mathrm{SK} = (\mathsf{sk}, \mathrm{VK}, \mathsf{vk}, \boldsymbol{\gamma})$ with $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \gamma_3)$ were generated by $\mathsf{LRS.KeyGen}$ and $\Sigma = (\mathsf{vk}, \mathsf{hk}^{(1)}, \mathsf{hk}^{(2)}, \mathsf{hk}^{(3)}, \pi, \sigma)$ is the output of $\mathsf{LRS.Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$, where $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$. We will show that it holds that $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma) = 1$. As $\mathsf{SPB.Hash}$ is deterministic, it holds for the hashes $\tilde{\mathsf{h}}^{(1)}, \tilde{\mathsf{h}}^{(2)}, \tilde{\mathsf{h}}^{(3)}$ computed by $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$ that $\tilde{\mathsf{h}}^{(i)} = \mathsf{h}^{(i)}$ (for $i = 1, 2, 3$), where the $\mathsf{h}^{(i)}$ are the hashes computed by $\mathsf{LRS.Sign}(\mathrm{SK}, \mathsf{m}, \mathsf{R})$. Also, it obviously holds that $\mathrm{VK} = \mathrm{VK}^{(1)}$. Now, notice further that by the correctness of SPB it holds that $\mathsf{SPB.Verify}(\mathsf{hk}^{(1)}, \mathsf{h}^{(1)}, \mathsf{ind}, \mathrm{VK}, \tau^{(1)}) = 1$. By the correctness of the commitment scheme Com, it holds that $\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}, \gamma_j) = 1$ for $j = 1, 2, 3$. Thus, $w = ((\mathsf{ind}, \mathrm{VK}, \tau^{(1)}, \boldsymbol{\gamma})), \emptyset, \emptyset, \emptyset)$ is a valid witness for the statement $x = (\mathsf{vk}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]})$. Consequently, by the correctness of NIWI it holds that $\mathsf{NIWI.Verify}(x, \pi) = 1$. Finally, by the correctness of Sig we get that $\mathsf{Sig.Verify}(\mathsf{vk}, (\mathsf{m}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}, \pi), \sigma) = 1$ and $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}, \Sigma)$ outputs 1.

6.2 Signature Size

For a signature $\Sigma = (\mathsf{vk}, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi, \sigma)$, the size of the signature component σ is $\mathsf{poly}(\lambda)$ and independent of the ring-size ℓ. By the efficiency property of SPB the sizes of the hashing keys $\mathsf{hk}^{(1)}, \mathsf{hk}^{(2)}, \mathsf{hk}^{(3)}$ is bounded by $\log(\ell)\cdot\mathsf{poly}(\lambda)$. Furthermore, for a statement $x = (\mathsf{vk}, (\mathsf{hk}^{(i)}, \tilde{\mathsf{h}}^{(i)})_{i\in[3]})$, the size of the verification circuit C_x is dominated $\mathsf{SPB.Verify}$, which by the efficiency property of SPB can be computed by a circuit of size $\log(\ell)\cdot\mathsf{poly}(\lambda)$. All other algorithms can be computed by circuits of size $\mathsf{poly}(\lambda)$ and independent of ℓ. Consequently, it holds that $|C_x| = \log(\ell)\cdot\mathsf{poly}(\lambda)$. By the efficiency property of the NIWI proof, it holds that $|\pi| = |C_x|\cdot\mathsf{poly}(\lambda) = \log(\ell)\cdot\mathsf{poly}(\lambda)$. All together, we can conclude that $|\Sigma| = \log(\ell)\cdot\mathsf{poly}(\lambda)$.

6.3 Unforgeability.

We will turn to showing that LRS is unforgeable. Basically, LRS inherits its unforgeability directly from Sig.

Theorem 3. *The ring signature scheme* LRS *is unforgeable, given that* NIWI *has perfect soundness,* SPB *is somewhere perfectly binding,* Com *is perfectly binding and* Sig *is unforgeable.*

The main idea of the proof is as follows. As all keys in the ring R chosen by the adversary are well formed, and further as Com is perfectly binding and NIWI is perfectly sound, the verification key vk used by the adversary $\mathcal{A}$ to produce a forge must be one of the keys committed to in one of the verification keys VK_i of R. Thus, we can leverage $\mathcal{A}$ to forge a signature under vk.

Proof. Let $\mathcal{A}$ be a PPT-adversary against the unforgeability experiment of LRS and let further $q = \mathsf{poly}(\lambda)$ be an upper bound on the number of key queries made by $\mathcal{A}$. Let $\mathsf{Exp}_{\mathsf{LRS\text{-}Unf}}$ be the unforgeability experiment of LRS and $\mathsf{Exp}_{\mathsf{EUF-CMA}}$ be the unforgeability experiment of Sig. We will construct a reduction $\mathcal{R}$ such that it holds that $\mathsf{Adv}_{\mathsf{Exp}_{\mathsf{EUF-CMA}}}(\mathcal{R}^{\mathcal{A}}) \geq \frac{1}{q}\cdot\mathsf{Adv}_{\mathsf{LRS\text{-}Unf}}(\mathcal{A})$. The reduction $\mathcal{R}$ is given as follows.

Reduction $\mathcal{R}^{\mathcal{A}}(\mathsf{vk}^*)$

- Guess an index $k^* \leftarrow_{\$} [q]$. On the k-th key query of $\mathcal{A}$, if $k \neq k^*$, answer the query as in $\mathsf{Exp}_{\mathsf{LRS\text{-}Unf}}$. For the k^*-th key query, proceed as follows.
  - For $j = 1, 2, 3$ compute $(\mathsf{com}_j^*, \gamma_j^*) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk}^*)$
  - Set $\boldsymbol{\gamma}^* \leftarrow (\gamma_j^*)_{j\in[3]}$
  - Output $\mathrm{VK}^* \leftarrow (\mathsf{com}_j^*)_{j\in[3]}$ and $\mathrm{SK}^* \leftarrow (\emptyset, \mathrm{VK}^*, \boldsymbol{\gamma}^*)$
- If $\mathcal{A}$ asks to corrupt VK^* abort.
- If $\mathcal{A}$ sends a signature query of the form $(\mathsf{m}, \mathrm{VK}^*, \mathsf{R})$, proceed as follows.
  - Parse $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$
  - Retrieve $\mathrm{SK}^* = (\emptyset, \mathrm{VK}^*, \mathsf{vk}^*, \boldsymbol{\gamma}^*)$
  - Parse $\mathrm{VK}^* = (\mathsf{com}_j^*)_{j\in[3]}$
  - Find an index $\mathsf{ind}^* \in [\ell]$ such that $\mathrm{VK}_{\mathsf{ind}^*} = \mathrm{VK}^*$

  - For $i = 1, 2, 3$ compute $(\mathsf{hk}^{(i)}, \mathsf{shk}^{(i)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}|, \mathsf{ind}^*)$ and $\mathsf{h}^{(i)} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}^{(i)}, \mathsf{R})$.
  - Compute $\tau^{(1)} \leftarrow \mathsf{SPB.Open}(\mathsf{hk}^{(1)}, \mathsf{shk}^{(1)}, \mathsf{R}, \mathsf{ind}^*)$
  - Set $x \leftarrow (\mathsf{vk}^*, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]})$
  - Set $w \leftarrow ((\mathsf{ind}^*, \mathrm{VK}^*, \tau^{(1)}, \boldsymbol{\gamma}), \emptyset, \emptyset, \emptyset)$.
  - Compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$.
  - Send $(\mathsf{m}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}, \pi)$ to the signing oracle and receive a signature σ.
  - Output $\Sigma \leftarrow (\mathsf{vk}^*, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi, \sigma)$

– Once $\mathcal{A}$ outputs a forge Σ^* for $(\mathsf{m}^*, \mathsf{R}^*)$, check if it is valid, that is in the query phase $\mathcal{A}$ has not requested a signature of m^* for any key in R^*, none of the keys in R^* has been corrupted and it holds that $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}^*, \Sigma^*) = 1$. If the forge is valid proceed, otherwise abort.
– Parse Σ^* as $\Sigma^* = (\tilde{\mathsf{vk}}, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi^*, \sigma^*)$.
– If $\tilde{\mathsf{vk}} = \mathsf{vk}^*$, output message $(\mathsf{m}^*, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}, \pi^*)$ and signature σ^*, otherwise abort.

First notice that unless $\mathcal{A}$ asks to corrupt VK^*, the experiment $\mathsf{Exp}_{\mathsf{LRS\text{-}Unf}}$ and the simulation of $\mathcal{R}$ are identically distributed from the view of $\mathcal{A}$. Therefore, from the view of $\mathcal{A}$ the index of the key VK^* is distributed uniformly at random. Consequently, with probability at least $1/q$ the adversary $\mathcal{A}$ does not trigger an abort by a corruption query to VK^*.

Assume now that $\mathcal{A}$ outputs a valid forge $\Sigma^* = (\tilde{\mathsf{vk}}, (\mathsf{hk}^{(i)})_{i\in[3]}, \pi, \sigma^*)$ for $(\mathsf{m}^*, \mathsf{R}^*)$ with $\mathsf{R}^* = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$. Let $\mathsf{h}^{(i)} = \mathsf{SPB.Hash}(\mathsf{hk}^{(i)}, \mathsf{R}^*)$ for $i = 1, 2, 3$. As $\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{m}^*, \Sigma^*) = 1$, it holds that $\mathsf{NIWI.Verify}((\tilde{\mathsf{vk}}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}), \pi) = 1$ and $\mathsf{Sig.Verify}(\tilde{\mathsf{vk}}, (\mathsf{m}^*, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}, \pi), \sigma^*) = 1$. Consequently, it holds by the perfect soundness of NIWI that $(\tilde{\mathsf{vk}}, (\mathsf{hk}^{(i)}, \mathsf{h}^{(i)})_{i\in[3]}) \in \mathcal{L}$. Thus there exists a witness $w = ((\mathsf{ind}^{(i)}, \mathrm{VK}^{(i)}, \tau^{(i)}, \boldsymbol{\gamma}^{(i)})_{i\in[3]}, \mathsf{vk}')$ where $\mathrm{VK}^{(i)} = (\mathsf{com}_1^{(i)}, \mathsf{com}_2^{(i)}, \mathsf{com}_3^{(i)})$ and $\boldsymbol{\gamma}^{(i)} = (\gamma_1^{(i)}, \gamma_2^{(i)}, \gamma_3^{(i)})$ for $i = 1, \ldots, 3$ such that it holds that

$$
\begin{aligned}
&\mathsf{SPB.Verify}(\mathsf{hk}^{(1)}, \mathsf{h}^{(1)}, \mathsf{ind}^{(1)}, \mathrm{VK}^{(1)}, \tau^{(1)}) = 1 \\
&\text{ and } \forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_j^{(1)}, \tilde{\mathsf{vk}}, \gamma_j^{(1)}) = 1 \\
&\text{ or} \\
&\mathsf{ind}^{(2)} \neq \mathsf{ind}^{(3)} \\
&\text{ and } \forall i \in \{2, 3\} : \mathsf{SPB.Verify}(\mathsf{hk}^{(i)}, \mathsf{h}^{(i)}, \mathsf{ind}^{(i)}, \mathrm{VK}^{(i)}, \tau^{(i)}) = 1 \\
&\text{ and } \mathsf{JointVerify}(\mathrm{VK}^{(2)}, \mathrm{VK}^{(3)}, \tilde{\mathsf{vk}}, \mathsf{vk}', \boldsymbol{\gamma}^{(2)}, \boldsymbol{\gamma}^{(3)}) = 1.
\end{aligned}
$$

Now recall that all the keys in R^* are honestly generated. That is, the expression $\mathsf{JointVerify}(\mathrm{VK}^{(2)}, \mathrm{VK}^{(3)}, \tilde{\mathsf{vk}}, \mathsf{vk}', \boldsymbol{\gamma}^{(2)}, \boldsymbol{\gamma}^{(3)}) = 1$ implies that either $\forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_j^{(2)}, \tilde{\mathsf{vk}}, \gamma_j^{(2)}) = 1$ or $\forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_j^{(3)}, \tilde{\mathsf{vk}}, \gamma_j^{(3)}) = 1$. Thus it must hold for an $i^* \in [3]$ both $\mathsf{SPB.Verify}(\mathsf{hk}^{(i^*)}, \mathsf{h}^{(i^*)}, \mathsf{ind}^{(i^*)}, \mathrm{VK}^{(i^*)}, \tau^{(i^*)}) = 1$ and $\forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_j^{(i^*)}, \tilde{\mathsf{vk}}, \gamma_j^{(i^*)}) = 1$.

Now, by the somewhere perfectly binding property of SPB it holds that $\mathrm{VK}^{(i^*)}$ is in the ring R^* and by the perfectly binding property of Com it holds that $\tilde{\mathsf{vk}}$ is the key that $\mathrm{VK}^{(i^*)} = (\mathsf{com}_1^{(i^*)}, \mathsf{com}_2^{(i^*)}, \mathsf{com}_3^{(i^*)})$ commits to.

Finally, observe that the index k^* of the key VK^* is uniformly random in the view of $\mathcal{A}$, thus it holds with probability at least $1/q$ that $\mathrm{VK}^{(i^*)} = \mathrm{VK}^*$ and therefore $\tilde{\mathsf{vk}} = \mathsf{vk}^*$. If this event happens, then $\mathcal{R}^{\mathcal{A}}$ outputs a valid forge σ^* for vk^*.

We conclude that $\mathsf{Adv}_{\mathsf{Exp}_{\mathsf{EUF-CMA}}}(\mathcal{R}^{\mathcal{A}}) \geq \frac{1}{q} \cdot \mathsf{Adv}_{\mathsf{LRS\text{-}Unf}}(\mathcal{A})$, which concludes the proof.

6.4 Linkable Anonymity.

We will now turn to establishing linkable anonymity of LRS.

Theorem 4. *The ring signature scheme* LRS *is linkably anonymous, given that* SPB *is index hiding,* Com *is computationally hiding and* NIWI *is computationally witness-indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT-adversary against the linkable anonymity of LRS. Assume that $\mathcal{A}$ makes at most $q = \mathsf{poly}(\lambda)$ key queries. Consider the following hybrids:

$\mathcal{H}_0$: This experiment is identical to the real experiment $\mathsf{Exp}_{\mathsf{LRS\text{-}Link}}$, except for the following modification. Before interacting with $\mathcal{A}$, the experiment guesses two key indices $i_0^*, i_1^* \in [q]$. Later, when $\mathcal{A}$ announces two indices i_0, i_1 in the challenge phase, the experiment aborts if $(i_0, i_1) \neq (i_0^*, i_1^*)$.

It follows immediately that $\mathsf{Adv}_{\mathcal{H}_0}(\mathcal{A}) \geq \frac{1}{q^2}\mathsf{Adv}_{\mathcal{H}_0'}(\mathcal{A})$. We will now show via a sequence of hybrids that show that $\mathsf{Adv}_{\mathcal{H}_0}(\mathcal{A})$ is negligible.

For convenience, let in the following hybrids VK_0 be the verification key at key-index i_0^* and VK_1 be the verification key at key-index i_1^*. We will briefly review how $\mathrm{VK}_0, \mathrm{VK}_1$ and Σ_0, Σ_1 are computed in $\mathcal{H}_0$. VK_β for $\beta = 0, 1$ is computed by

- $(\mathsf{vk}_\beta, \mathsf{sk}_\beta) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
- For $i = 1, 2, 3$ compute $(\mathsf{com}_{\beta,j}, \gamma_{\beta,j}) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk}_\beta)$
- $\boldsymbol{\gamma}_\beta \leftarrow (\gamma_{\beta,j})_{j\in[3]}$
- $\mathrm{VK}_\beta \leftarrow (\mathsf{com}_{\beta,j})_{j\in[3]}$ and $\mathrm{SK}_\beta \leftarrow (\mathsf{sk}_\beta, \mathrm{VK}_\beta, \mathsf{vk}_\beta, \boldsymbol{\gamma}_\beta)$.

Moreover, let Σ_0 be challenge signature of m_0 under R_0 and Σ_1 be the challenge signature of m_1 under R_1. To facilitate notation let $\mathsf{ind}_{c,d}$ be the index of VK_d in ring R_c^* for $c, d \in \{0, 1\}$ (these 4 indices will be used frequently in the proof). We will also briefly review how Σ_0 and Σ_1 are computed.

The signature Σ_β for $\beta = 0, 1$ is computed by

- For $i = 1, 2, 3$ compute $(\mathsf{hk}_\beta^{(i)}, \mathsf{shk}_\beta^{(i)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_\beta|, \mathsf{ind}_{\beta,\beta})$ and $\mathsf{h}_\beta^{(i)} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_\beta^{(i)}, \mathsf{R}_\beta)$
- $\tau_\beta^{(1)} \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_\beta^{(1)}, \mathsf{shk}_\beta^{(1)}, \mathsf{R}, \mathsf{ind}_{\beta,\beta})$

- $x_\beta \leftarrow (\mathsf{vk}_\beta, (\mathsf{hk}_\beta^{(i)}, \mathsf{h}_\beta^{(i)})_{i\in[3]})$
- $w_\beta \leftarrow ((\mathsf{ind}_{\beta,\beta}, \mathrm{VK}_\beta, \tau_\beta^{(1)}, \boldsymbol{\gamma}_\beta), \emptyset, \emptyset, \emptyset)$.
- $\pi_\beta \leftarrow \mathsf{NIWI.Prove}(x_\beta, w_\beta)$
- $\sigma_\beta \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_\beta, (\mathsf{m}_\beta, (\mathsf{hk}_\beta^{(i)}, \mathsf{h}_\beta^{(i)})_{i\in[3]}, \pi_\beta))$
- $\Sigma_\beta \leftarrow (\mathsf{vk}_\beta, (\mathsf{hk}_\beta^{(i)})_{i\in[3]}, \pi_\beta, \sigma_\beta)$.

This concludes the review of the structure of $\mathrm{VK}_0, \mathrm{VK}_1$ and Σ_0, Σ_1. Now consider the following hybrids.

$\mathcal{H}_0^0$: This is the experiment $\mathcal{H}_0$ with challenge bit $b = 0$.

$\mathcal{H}_1$: This experiment is identical to $\mathcal{H}_0^0$, except for the following conceptual change. We will compute VK_0 and VK_1 simultaneously before the experiment begins.

- $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
- $(\mathsf{vk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
- For $i = 1, 2, 3$ compute $(\mathsf{com}_{0,j}, \gamma_{0,j}) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk}_0)$
- For $i = 1, 2, 3$ compute $(\mathsf{com}_{1,j}, \gamma_{1,j}) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk}_1)$
- $\boldsymbol{\gamma}_0 \leftarrow (\gamma_{0,j})_{j\in[3]}$
- $\boldsymbol{\gamma}_0 \leftarrow (\gamma_{0,j})_{j\in[3]}$
- $\mathrm{VK}_0 \leftarrow (\mathsf{com}_{0,j})_{j\in[3]}$ and $\mathrm{SK}_0 \leftarrow (\mathsf{sk}_0, \mathrm{VK}_0, \mathsf{vk}_0, \boldsymbol{\gamma}_0)$.
- $\mathrm{VK}_0 \leftarrow (\mathsf{com}_{0,j})_{j\in[3]}$ and $\mathrm{SK}_0 \leftarrow (\mathsf{sk}_0, \mathrm{VK}_0, \mathsf{vk}_0, \boldsymbol{\gamma}_0)$.

That is, in the computation of VK_0 and VK_1 we first choose vk_0 and vk_1 before computing anything else. Looking ahead, this will be important later on when we change the roles of vk_0 and vk_1. This change is only conceptual and therefore $\mathcal{H}_0^0$ and $\mathcal{H}_1$ are identically distributed from the view of $\mathcal{A}$.

$\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except that we compute $\mathsf{hk}_0^{(2)}, \mathsf{hk}_0^{(3)}, \mathsf{hk}_1^{(2)}, \mathsf{hk}_1^{(3)}$ as follows:

$$
\begin{aligned}
(\mathsf{hk}_0^{(2)}, \mathsf{shk}_0^{(2)}) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,0}) \\
(\mathsf{hk}_0^{(3)}, \mathsf{shk}_0^{(3)}) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,1}) \\
(\mathsf{hk}_1^{(2)}, \mathsf{shk}_1^{(2)}) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_1|, \mathsf{ind}_{1,0}) \\
(\mathsf{hk}_1^{(3)}, \mathsf{shk}_1^{(3)}) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_1|, \mathsf{ind}_{1,1}).
\end{aligned}
$$

That is, we move the binding index of $\mathsf{hk}_0^{(3)}$ from $\mathsf{ind}_{0,0}$ to $\mathsf{ind}_{0,1}$ and of $\mathsf{hk}_1^{(2)}$ from $\mathsf{ind}_{1,1}$ to $\mathsf{ind}_{1,0}$. $\mathsf{hk}_0^{(2)}$ and $\mathsf{hk}_1^{(3)}$ are computed as before and only stated here for convenience. The secret keys $\mathsf{shk}_0^{(2)}, \mathsf{shk}_0^{(3)}, \mathsf{shk}_1^{(2)}, \mathsf{shk}_1^{(3)}$ are not needed to compute information which is used in the witnesses w_0 and w_1 in the experiments $\mathcal{H}_1$ and $\mathcal{H}_2$. Therefore, we can argue that $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable via the index-hiding property of SPB.

$\mathcal{H}_3$: In this hybrid we also compute

$$
\begin{aligned}
\tau_0^{(2)} &\leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0^{(2)}, \mathsf{shk}_0^{(2)}, \mathrm{VK}_0, \mathsf{ind}_{0,0}) \\
\tau_0^{(3)} &\leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0^{(3)}, \mathsf{shk}_0^{(3)}, \mathrm{VK}_0, \mathsf{ind}_{0,1})
\end{aligned}
$$

in the computation of Σ_0 and

$$\tau_1^{(2)} \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1^{(2)}, \mathsf{shk}_1^{(2)}, \mathrm{VK}_0, \mathsf{ind}_{1,0})$$
$$\tau_1^{(3)} \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1^{(3)}, \mathsf{shk}_1^{(3)}, \mathrm{VK}_0, \mathsf{ind}_{1,1})$$

in the computation of Σ_1. The values $\tau_0^{(2)}, \tau_0^{(3)}, \tau_1^{(2)}, \tau_1^{(3)}$ are not used further in $\mathcal{H}_3$. Thus, this modification is only conceptual and we get that $\mathcal{H}_2$ and $\mathcal{H}_3$ are identically distributed from the view of $\mathcal{A}$.

$\mathcal{H}_4$: In this hybrid we switch the witnesses w_0 and w_1 used for the computation of Σ_0 and Σ_1. Specifically, instead of computing

$$w_0 \leftarrow ((\mathsf{ind}_{0,0}, \mathrm{VK}_0, \tau_0^{(1)}, \boldsymbol{\gamma}_0), \emptyset, \emptyset, \emptyset)$$
$$w_1 \leftarrow ((\mathsf{ind}_{1,1}, \mathrm{VK}_1, \tau_1^{(1)}, \boldsymbol{\gamma}_1), \emptyset, \emptyset, \emptyset)$$

we compute

$$w_0 \leftarrow (\emptyset, (\mathsf{ind}_{0,0}, \mathrm{VK}_0, \tau_0^{(2)}, \boldsymbol{\gamma}_0), (\mathsf{ind}_{0,1}, \mathrm{VK}_1, \tau_0^{(3)}, \boldsymbol{\gamma}_1), \mathsf{vk}_1)$$
$$w_1 \leftarrow (\emptyset, (\mathsf{ind}_{1,0}, \mathrm{VK}_0, \tau_1^{(2)}, \boldsymbol{\gamma}_0), (\mathsf{ind}_{1,1}, \mathrm{VK}_1, \tau_1^{(3)}, \boldsymbol{\gamma}_1), \mathsf{vk}_0).$$

First notice that the new w_0 is also a witness for the statement $x_0 = (\mathsf{vk}_0, (\mathsf{hk}_0^{(i)}, \mathsf{h}_0^{(i)})_{i\in[3]})$ as $\mathsf{JointVerify}(\mathrm{VK}_0, \mathrm{VK}_1, \mathsf{vk}_0, \mathsf{vk}_1, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1) = 1$ holds. Recall that this predicate holds if at least two out of the six commitments correctly open to vk_0, at most one commitment does not open correctly, and the remaining commitments open to vk_1. Likewise, the witness w_1 is a witness for the statement $x_1 = (\mathsf{vk}_1, (\mathsf{hk}_1^{(i)}, \mathsf{h}_1^{(i)})_{i\in[3]})$ as $\mathsf{JointVerify}(\mathrm{VK}_0, \mathrm{VK}_1, \mathsf{vk}_1, \mathsf{vk}_0, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1) = 1$. We can argue via the computational witness-indistinguishability of NIWI that $\mathcal{H}_3$ and $\mathcal{H}_4$ are computationally indistinguishable form the view of $\mathcal{A}$.

In the following hybrids $\mathcal{H}_5, \ldots, \mathcal{H}_{17}$, we will simultaneously modify $\mathrm{VK}_0 = (\mathsf{com}_{0,1}, \mathsf{com}_{0,2}, \mathsf{com}_{0,3})$ and $\mathrm{VK}_1 = (\mathsf{com}_{1,1}, \mathsf{com}_{1,2}, \mathsf{com}_{1,3})$ such that VK_0 commits to vk_1 and VK_1 commits to vk_0. In other words, we will switch the roles of VK_0 and VK_1. The configuration in each hybrid is given in table 1. Recall that the joint verfication algorithm $\mathsf{JointVerify}$ tolerates one incorrect/missing unveal. In order to use the hiding property of Com, in each hybrid we will *forget* the unveal γ for (at most) one commitment. The commitment for which the unveal is missing is given in a grey box. As a brief explanation, in $\mathcal{H}_4$ all unveils γ are present. In $\mathcal{H}_5$ we forget the unveal $\gamma_{0,1}$ of the commitment $\mathsf{com}_{0,1}$, i.e. we set $\gamma_{0,1} \leftarrow \emptyset$. Indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_5$ follows by the witness indistinguishability of NIWI. In $\mathcal{H}_6$, we change $\mathsf{com}_{0,1}$ into a commitment of vk_1 and can argue indistinguishability of $\mathcal{H}_5$ and $\mathcal{H}_6$ via the hiding property of Com. In $\mathcal{H}_7$ we erase the unveal information $\gamma_{1,1}$ of $\mathsf{com}_{1,1}$ instead of $\gamma_{0,0}$ indistinguishability of $\mathcal{H}_6$ and $\mathcal{H}_7$ follows again by the witness-indistinguishability of NIWI. The remaining steps follow analogously, observing that in each row of table 1 it holds *both* $\mathsf{JointVerify}(\mathrm{VK}_0, \mathrm{VK}_1, \mathsf{vk}_0, \mathsf{vk}_1, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1) = 1$ and $\mathsf{JointVerify}(\mathrm{VK}_0, \mathrm{VK}_1, \mathsf{vk}_1, \mathsf{vk}_0, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1) = 1$.

Hybrid	$\mathsf{com}_{0,1}$	$\mathsf{com}_{0,2}$	$\mathsf{com}_{0,3}$	$\mathsf{com}_{1,1}$	$\mathsf{com}_{1,2}$	$\mathsf{com}_{1,3}$
$\mathcal{H}_4$	vk_0	vk_0	vk_0	vk_1	vk_1	vk_1
$\mathcal{H}_5$	vk_0	vk_0	vk_0	vk_1	vk_1	vk_1
$\mathcal{H}_6$	vk_1	vk_0	vk_0	vk_1	vk_1	vk_1
$\mathcal{H}_7$	vk_1	vk_0	vk_0	vk_1	vk_1	vk_1
$\mathcal{H}_8$	vk_1	vk_0	vk_0	vk_0	vk_1	vk_1
$\mathcal{H}_9$	vk_1	vk_0	vk_0	vk_0	vk_1	vk_1
$\mathcal{H}_{11}$	vk_1	vk_1	vk_0	vk_0	vk_1	vk_1
$\mathcal{H}_{11}$	vk_1	vk_1	vk_0	vk_0	vk_1	vk_1
$\mathcal{H}_{12}$	vk_1	vk_1	vk_0	vk_0	vk_0	vk_1
$\mathcal{H}_{13}$	vk_1	vk_1	vk_0	vk_0	vk_0	vk_1
$\mathcal{H}_{14}$	vk_1	vk_1	vk_1	vk_0	vk_0	vk_1
$\mathcal{H}_{15}$	vk_1	vk_1	vk_1	vk_0	vk_0	vk_1
$\mathcal{H}_{16}$	vk_1	vk_1	vk_1	vk_0	vk_0	vk_0
$\mathcal{H}_{17}$	vk_1	vk_1	vk_1	vk_0	vk_0	vk_0

Table 1. The hybrids

$\mathcal{H}_{18}$: Identical to $\mathcal{H}_{17}$, except that we compute $\mathsf{hk}_0^{(1)}$ and $\mathsf{hk}_1^{(1)}$ differently. Instead of computing $\mathsf{hk}_0^{(1)}$ and $\mathsf{hk}_1^{(1)}$ by

$$(\mathsf{hk}_0^{(1)}, \mathsf{shk}_0^{(1)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,0})$$
$$(\mathsf{hk}_1^{(1)}, \mathsf{shk}_1^{(1)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_1|, \mathsf{ind}_{1,1})$$

we compute

$$(\mathsf{hk}_0^{(1)}, \mathsf{shk}_0^{(1)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,1})$$
$$(\mathsf{hk}_1^{(1)}, \mathsf{shk}_1^{(1)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{1,0}).$$

That is, we move the binding index of $\mathsf{hk}_0^{(1)}$ from $\mathsf{ind}_{0,0}$ to $\mathsf{ind}_{0,1}$ and of $\mathsf{hk}_1^{(1)}$ from $\mathsf{ind}_{1,1}$ to $\mathsf{ind}_{1,0}$. The secret keys $\mathsf{shk}_0^{(1)}, \mathsf{shk}_1^{(1)}$ are not needed to compute information which is used in the witnesses w_0 and w_1 in $\mathcal{H}_{17}$ and $\mathcal{H}_{18}$. Therefore, we can argue that $\mathcal{H}_{17}$ and $\mathcal{H}_{18}$ are computationally indistinguishable via the index-hiding property of SPB.

In the remaining hybrids, we essentially mirror the modification in hybrids $\mathcal{H}_1, \ldots, \mathcal{H}_4$.

$\mathcal{H}_{19}$ (mirroring $\mathcal{H}_4$): The same as $\mathcal{H}_{18}$, except that we switch the witnesses w_0 and w_1 used for the computation of Σ_0 and Σ_1. Specifically, instead of computing

$$w_0 \leftarrow (\emptyset, (\mathsf{ind}_{0,0}, \mathrm{VK}_0, \tau_0^{(2)}, \boldsymbol{\gamma}_0), (\mathsf{ind}_{0,1}, \mathrm{VK}_1, \tau_0^{(3)}, \boldsymbol{\gamma}_1), \mathsf{vk}_1)$$
$$w_1 \leftarrow (\emptyset, (\mathsf{ind}_{1,0}, \mathrm{VK}_0, \tau_1^{(2)}, \boldsymbol{\gamma}_0), (\mathsf{ind}_{1,1}, \mathrm{VK}_1, \tau_1^{(3)}, \boldsymbol{\gamma}_1), \mathsf{vk}_0)$$

we compute

$$w_0 \leftarrow ((\mathsf{ind}_{0,1}, \mathrm{VK}_1, \tau_0^{(1)}, \boldsymbol{\gamma}_1), \emptyset, \emptyset, \emptyset)$$
$$w_1 \leftarrow ((\mathsf{ind}_{1,0}, \mathrm{VK}_0, \tau_1^{(1)}, \boldsymbol{\gamma}_0), \emptyset, \emptyset, \emptyset).$$

First notice that the new w_0 is also a witness for the statement $x_0 = (\mathsf{vk}_0, (\mathsf{hk}_0^{(i)}, \mathsf{h}_0^{(i)})_{i\in[3]})$ as it holds $\forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_{1,j}^{(1)}, \mathsf{vk}_0, \gamma_{1,j}^{(1)}) = 1$. Likewise, the witness w_1 is a witness for the statement $x_1 = (\mathsf{vk}_1, (\mathsf{hk}_1^{(i)}, \mathsf{h}_1^{(i)})_{i\in[3]})$ as it holds $\forall j \in [3] : \mathsf{Com.Verify}(\mathsf{com}_{0,j}^{(1)}, \mathsf{vk}_1, \gamma_{0,j}^{(1)}) = 1$. We can argue via the computational witness-indistinguishability of NIWI that $\mathcal{H}_{18}$ and $\mathcal{H}_{19}$ are computationally indistinguishable form the view of $\mathcal{A}$.

$\mathcal{H}_{20}$ (mirroring $\mathcal{H}_3$): In this hybrid we drop the computation of $\tau_0^{(2)}, \tau_0^{(3)}$ in the computation of Σ_0 and we also drop the computation of $\tau_1^{(2)}, \tau_1^{(3)}$ in the computation of Σ_1. The values $\tau_0^{(2)}, \tau_0^{(3)}, \tau_1^{(2)}, \tau_1^{(3)}$ are not used further in $\mathcal{H}_{19}$. Thus, this modification is only conceptual and we get that $\mathcal{H}_{19}$ and $\mathcal{H}_{20}$ are identically distributed from the view of $\mathcal{A}$.

$\mathcal{H}_{21}$ (mirroring $\mathcal{H}_2$): Identical to $\mathcal{H}_{20}$, except that we compute $\mathsf{hk}_0^{(2)}$, $\mathsf{hk}_0^{(3)}$, $\mathsf{hk}_1^{(2)}$, $\mathsf{hk}_1^{(3)}$ as follows:

$$(\mathsf{hk}_0^{(2)}, \mathsf{shk}_0^{(2)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,1})$$
$$(\mathsf{hk}_0^{(3)}, \mathsf{shk}_0^{(3)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{0,1})$$
$$(\mathsf{hk}_1^{(2)}, \mathsf{shk}_1^{(2)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{1,0})$$
$$(\mathsf{hk}_1^{(3)}, \mathsf{shk}_1^{(3)}) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |\mathsf{R}_0|, \mathsf{ind}_{1,0}).$$

That is, we move the binding index of $\mathsf{hk}_0^{(2)}$ from $\mathsf{ind}_{0,0}$ to $\mathsf{ind}_{0,1}$ and of $\mathsf{hk}_1^{(3)}$ from $\mathsf{ind}_{1,1}$ to $\mathsf{ind}_{1,0}$. $\mathsf{hk}_0^{(3)}$ and $\mathsf{hk}_1^{(2)}$ are computed as before and only stated here for convenience. The secret keys $\mathsf{shk}_0^{(2)}, \mathsf{shk}_0^{(3)}, \mathsf{shk}_1^{(2)}, \mathsf{shk}_1^{(3)}$ are not needed to compute information which is used in the witnesses w_0 and w_1 in the experiments $\mathcal{H}_{20}$ and $\mathcal{H}_{21}$. Therefore, we can argue that $\mathcal{H}_{20}$ and $\mathcal{H}_{21}$ are computationally indistinguishable via the index-hiding property of SPB.

$\mathcal{H}_{22}$ (mirroring $\mathcal{H}_1$): Identical to $\mathcal{H}_{21}$ except that we compute VK_0 and VK_1 as follows. We compute VK_β for $\beta = 0, 1$ is computed by

- $(\mathsf{vk}_{1-\beta}, \mathsf{sk}_{1-\beta}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
- For $i = 1, 2, 3$ compute $(\mathsf{com}_{\beta,j}, \gamma_{\beta,j}) \leftarrow \mathsf{Com.Commit}(1^\lambda, \mathsf{vk}_{1-\beta})$
- $\boldsymbol{\gamma}_\beta \leftarrow (\gamma_{\beta,j})_{j\in[3]}$
- $\mathrm{VK}_\beta \leftarrow (\mathsf{com}_{\beta,j})_{j\in[3]}$ and $\mathrm{SK}_\beta \leftarrow (\mathsf{sk}_{1-\beta}, \mathrm{VK}_\beta, \mathsf{vk}_{1-\beta}, \boldsymbol{\gamma}_\beta)$.

In $\mathcal{H}_{21}$ we have computed VK_0 and VK_1 simultaneously. However, as the computation of VK_0 and VK_1 has no shared information, we can compute them independently of one another. Thus, this modification is only conceptual and we get that from the view of $\mathcal{A}$ $\mathcal{H}_{21}$ and $\mathcal{H}_{22}$ are identically distributed.

We now observe that from the view of $\mathcal{A}$, $\mathcal{H}_{22}$ is identically distributed to $\mathcal{H}_0^1$, i.e. $\mathcal{H}_0$ with challenge bit $b = 1$. Consequently, we get that

$$\mathsf{Adv}_{\mathcal{H}_0}(\mathcal{A}) = |\Pr[\mathcal{H}_0^0(\mathcal{A}) = 1 \ - \Pr[\mathcal{H}_0^1(\mathcal{A}) = 1]| < \mathsf{negl}(\lambda),$$

which concludes the proof.

6.5 Linkability.

Before we provide the proof of linkability of the scheme LRS we need the following lemma about the algorithm valid $\mathsf{JointVerify}$. For convenience, we provide the definition of the $\mathsf{JointVerify}$ function again.

$\mathsf{JointVerify}(\mathrm{VK}, \mathrm{VK}', \mathsf{vk}, \mathsf{vk}', \boldsymbol{\gamma}, \boldsymbol{\gamma}')$:
- Parse $\mathrm{VK} = (\mathsf{com}_j)_{j\in[3]}$ and $\mathrm{VK}' = (\mathsf{com}'_j)_{j\in[3]}$
- Parse $\boldsymbol{\gamma} = (\gamma_j)_{j\in[3]}$, $\boldsymbol{\gamma}' = (\gamma'_j)_{j\in[3]}$
- Compute $s \leftarrow \sum_{j=1}^{3}(\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}, \gamma_j) + \mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}, \gamma'_j))$
- Compute $s' \leftarrow \sum_{j=1}^{3}(\mathsf{Com.Verify}(\mathsf{com}_j, \mathsf{vk}', \gamma_j) + \mathsf{Com.Verify}(\mathsf{com}'_j, \mathsf{vk}', \gamma'_j))$
- If it holds that $s \geq 2$ and $s' \geq 2$ and $s + s' \geq 5$ output 1, otherwise 0.

Before we proceed we will introduce some terminology that will facilitate notation.

Definition 11. *We say that a key* $\mathrm{VK} = (\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ *yields an element* x*, if there exist* $\gamma_1, \gamma_2, \gamma_3$ *such that* $\mathsf{Com.Verify}(\mathsf{com}_j, x, \gamma_j) = 1$ *for* $i = 1, 2, 3$*. We say that a pair of keys* $(\mathrm{VK}, \mathrm{VK}')$ *yield* x*, if there exist* y *and* $\boldsymbol{\gamma}, \boldsymbol{\gamma}'$ *such that* $\mathsf{JointVerify}(\mathrm{VK}, \mathrm{VK}', x, y, \boldsymbol{\gamma}, \boldsymbol{\gamma}') = 1$*. Furthermore, we say that a ring* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$ *yields a set* $\mathcal{S}$*, if for every* $x \in \mathcal{S}$ *there exists a* $\mathrm{VK}_i \in \mathsf{R}$ *such that* VK_i *yields* x *or there exist distinct* $\mathrm{VK}_{i_1}, \mathrm{VK}_{i_2}$ *such that* $(\mathrm{VK}_{i_1}, \mathrm{VK}_{i_2})$ *yield* x*. Say that* $\mathrm{VK}^* \in \mathsf{R}$ *has a partner in* R*, if there exists a* $\tilde{\mathrm{VK}} \in \mathsf{R}$ *such that* $(\mathrm{VK}^*, \tilde{\mathrm{VK}})$ *yield an element in* $\mathcal{S}$*.*

In abuse of notation, we also say that VK is of the form $(\mathsf{com}(x), \mathsf{com}(y), \mathsf{com}(z))$, if one of the commitments in VK commits to x, one to y and one to z, where we allow x, y, z to be the same.

Lemma 1. *Assume that a ring* $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$ *yields a set* $\mathcal{S}$*. Then it holds that* $|\mathcal{S}| \leq |\mathsf{R}|$*.*

In other words, a ring of size n yields at most n distinct elements.

Proof. We will proof the lemma by induction over the size of R. For the base case of $|\mathcal{V}| = 2$, it follows immediately from the definition of $\mathsf{JointVerify}$ that R yields at most 2 elements. Assume that the assertion holds for rings of size at most $n - 1$. We will now show that it also holds for rings of size n. Let therefore R be any ring of size n. We will distinguish three cases.

Case 1: In this case R contains a VK^* of the form $(\mathsf{com}(x), \mathsf{com}(x), \mathsf{com}(x))$. Let $\mathcal{S}'$ be the set yielded by $\mathsf{R}\backslash\{\mathrm{VK}^*\}$. Assume that $\mathsf{R}\backslash\{\mathrm{VK}^*\}$ and VK^* together yield k elements outside of $\mathcal{S}'$. Call the set of these elements $\mathcal{T}$. We will show that there exists a sub-ring $\mathsf{R}' \subseteq \mathsf{R}$ of size $n-k$ which yields $\mathcal{S}'$. As $|\mathsf{R}'| \leq n-1$ we will conclude by the induction hypothesis that $|\mathcal{S}'| \leq |\mathsf{R}'| \leq n-k$, which in turn will imply that $|\mathcal{S}| \leq |\mathcal{S}'| + |\mathcal{T}| \leq n-k+k = n$.
For all $t \in \mathcal{T}$ with $t \neq x$ there must exist a $\mathrm{VK}_t^* \in \mathsf{R}\backslash\{\mathrm{VK}^*\}$ such that $(\mathrm{VK}^*, \mathrm{VK}_t^*)$ yield t. But this means that VK_t^* is of the form $(\mathsf{com}(t), \mathsf{com}(t), *)$, and this implies that VK_t^* does not have a partner in $\mathsf{R}\backslash\{\mathrm{VK}^*\}$, as otherwise it would hold $t \in \mathcal{S}'$. Moreover, for $t' \neq t$ it we immediately get by the above that $\mathrm{VK}_{t'}^* \neq \mathrm{VK}_t^*$. Thus, we can remove at least $k-1$ elements from $\mathsf{R}\backslash\{\mathrm{VK}^*\}$ (i.e. all VK_t^* corresponding to the elements in $\mathcal{T}\backslash\{x\}$) and obtain a ring R' of size $n-k$ while guaranteeing that the resulting ring still yields $\mathcal{S}'$. Using the induction hypothesis we conclude that $|\mathcal{S}'| \leq |\mathsf{R}'| \leq n-k$ and therefore $\mathcal{S} \leq n$.

Case 2: In this case R does not contain any VK of the form $(\mathsf{com}(x), \mathsf{com}(x), \mathsf{com}(x))$, but it does contain a VK^* of the form $(\mathsf{com}(x), \mathsf{com}(x), \mathsf{com}(y))$. Let $\mathcal{S}'$ be the set yielded by $\mathsf{R}\backslash\{\mathrm{VK}^*\}$. Again assume that $\mathsf{R}\backslash\{\mathrm{VK}^*\}$ and VK^* yield k elements outside of $\mathcal{S}'$ and call this set $\mathcal{T}$. As in case 1 we will show that there exists a sub-ring $\mathsf{R}' \subseteq \mathsf{R}$ of size $n-k$ which yields $\mathcal{S}'$. As $|\mathsf{R}'| \leq n-1$ we will conclude by the induction hypothesis that $|\mathcal{S}'| \leq |\mathsf{R}'| \leq n-k$, which in turn will imply that $|\mathcal{S}| \leq |\mathcal{S}'| + |\mathcal{T}| \leq n-k+k = n$.
For every $t \in \mathcal{T}$ different from x and y there exists a $\mathrm{VK}_t^* \in \mathsf{R}\backslash\{\mathrm{VK}^*\}$ such that VK_t^* and VK^* yield t. Such a VK_t^* must be of the form $\mathrm{VK}_t^* = (\mathsf{com}(t), \mathsf{com}(t), \mathsf{com}(x))$. Consequently, VK_t^* cannot have a partner in $\mathsf{R}\backslash\{\mathrm{VK}^*\}$, as this would imply that $t \in \mathcal{S}'$. For two distinct $t' \neq t \in \mathcal{T}$, it must hold that $\mathrm{VK}_t^* \neq \mathrm{VK}_{t'}^*$ by the above.
If $y \notin \mathcal{T}$ we are done, and will be able to argue as in case 1. If $y \in \mathcal{T}$, then there exists a $\mathrm{VK}_y^* \in \mathsf{R}\backslash\{\mathrm{VK}^*\}$ such that VK_y^* and VK^* yield y. But this means that VK_y^* must be of the form $(\mathsf{com}(x), \mathsf{com}(y), *)$. If VK_y^* has no partner in $\mathsf{R}\backslash\{\mathrm{VK}^*\}$ we are done and can argue as before. If VK_y^* has a partner in $\mathsf{R}\backslash\{\mathrm{VK}_y^*\}$ then it holds that $x \in \mathcal{S}'$, as $y \notin \mathcal{S}'$ (as it is in $\mathcal{T}$). Consequently, in any of these cases we can remove $k-1$ elements from $\mathsf{R}\backslash\{\mathrm{VK}^*\}$ and obtain a ring R' of size $n-k$ which yields $\mathcal{S}'$. Using the induction hypothesis we conclude that $|\mathcal{S}'| \leq |\mathsf{R}'| \leq n-k$ and therefore $\mathcal{S} \leq n$.

Case 3: R only contains VK of the form $(\mathsf{com}(x), \mathsf{com}(y), \mathsf{com}(z))$ for distinct x, y, z. None of these VK can be partnered and it immediately follows that $\mathcal{S} = \emptyset$.

This concludes the proof.

We will now show that our scheme is perfectly linkable.

Theorem 5. *The ring signature scheme* LRS *is perfectly linkable, given* SPB *is somewhere perfectly binding,* Com *is perfectly binding and* NIWI *has perfect soundness.*

Proof. Assume that the linking adversary $\mathcal{A}$ outputs a ring $\mathsf{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_\ell)$ of ℓ keys. Then it holds by Lemma 11 that R yields at most ℓ distinct keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell$. Assume that $\mathcal{A}$ outputs $\ell+1$ valid message-signature pairs (m_1, Σ_1), $\ldots$, $(\mathsf{m}_{\ell+1}, \Sigma_{\ell+1})$, where $\Sigma_i = (\tilde{\mathsf{vk}}_i, *, *, *)$. Then by the perfect soundness of NIWI, the somewhere perfectly binding property of SPB and the perfect binding property of Com it holds that each $\tilde{\mathsf{vk}}_i$ must be yieldable from one or two keys in R. But this means that $\tilde{\mathsf{vk}}_i$ is in the set $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell\}$. Since this set contains only ℓ elements, at least two of the Σ_i must link.

6.6 Non-Framability.

We will now show that LRS is non-framable.

Theorem 6. *Given that* Sig *is unforgeable,* Com *is perfectly binding and* NIWI *is perfectly sound, the scheme* LRS *has non-framability.*

The idea of the proof is simple. The only way the adversary can win the experiment is by producing a signature Sig^* which links to one of the honest keys. In order to achieve this however, the adversary must forge a signature under the key to which this key commits to, which contradicts the unforgeability of the underlying signature scheme.

Proof. Let $\mathcal{A}$ be a PPT adversary against the non-framability of LRS. Assume that $\mathcal{A}$ makes at most $q = \mathsf{poly}(\lambda)$ key queries. Consider the following hybrids.

$\mathcal{H}_0$: This is the real experiment.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except for the following modification. Let $\mathrm{VK}_1, \ldots, \mathrm{VK}_q$ be the keys generated by the experiment, where VK_i commits to a key vk_i of Sig. If in phase 1 the adversary $\mathcal{A}$ outputs a valid message-signature pair $(\mathsf{m}^*, \Sigma^* = (\tilde{\mathsf{vk}}, (\mathsf{hk}_h^*)_{j\in[3]}, \pi^*, \sigma^*))$ such that $\tilde{\mathsf{vk}} = \mathsf{vk}_i$ for an uncorrupted VK_i and $(\mathsf{m}, \mathrm{VK}_i, *)$ has not been queried to the signing oracle, then the experiment aborts and outputs 0.

We will first show that $|\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]| < \mathsf{negl}(\lambda)$ given that Sig is existentially unforgeable. Let $F(\mathcal{A})$ be the event that $\mathcal{A}$ outputs a signature $\Sigma^* = (\mathsf{vk}^*, (\mathsf{hk}_h)_{j\in[3]}, \pi)$ such that $\mathsf{vk}^* = \mathsf{vk}_i$ for an uncorrupted VK_i. Clearly, conditioned on $\neg F(\mathcal{A})$ both experiments are identically distributed. Assume therefore towards contradiction that $\Pr[F(\mathcal{A})] \geq \epsilon$ for a non-negligible ϵ. We will sketch a reduction $\mathcal{R}$ such that $\mathcal{R}^{\mathcal{A}}$ breaks the unforgeability of Sig with advantage ϵ. $\mathcal{R}$ first guesses an index $i^* \in [q]$ and uses its own input vk^* to generate the key VK_{i^*}. If $\mathcal{A}$ requests a signature of a message m under VK_{i^*} and ring $\mathcal{R}$, $\mathcal{R}$ computes the signature in the same way as in $\mathcal{H}_0$, but uses its own signing oracle with input $(\mathsf{m}, (\mathsf{hk}^{(i)}, \tilde{\mathsf{h}}^{(i)})_{i\in[3]}, \pi)$ to obtain the signature σ. Once $\mathcal{A}$ outputs (m^*, Σ^*) with $(\mathsf{vk}^*, (\mathsf{hk}_h^*)_{j\in[3]}, \pi^*, \sigma^*))$, $\mathcal{R}$ checks if $\tilde{\mathsf{vk}} = \mathsf{vk}^*$, and if so outputs $((\mathsf{m}^*, (\mathsf{hk}_h^*)_{j\in[3]}, \pi^*), \sigma^*)$.

Clearly, the index i^* is uniformly random in the view of $\mathcal{A}$ and therefore the event $\tilde{\mathsf{vk}} = \mathsf{vk}^*$ happens with probability at least $\frac{1}{q}$. Consequently, it holds that

$$\mathsf{Adv}_{\mathsf{EUF-CMA}}(\mathcal{R}^{\mathcal{A}}) \geq \frac{1}{q} \cdot \Pr[F(\mathcal{A})] \geq \frac{\epsilon}{q},$$

which contradicts the unforgeability of Sig.

Finally notice that in $\mathcal{H}_1$ the advantage of $\mathcal{A}$ is 0: Due to the perfect binding property of Com and the perfect soundness of NIWI, any signature $\Sigma^\dagger$ that $\mathcal{A}$ generates must use one of the $\mathsf{vk}_1, \ldots, \mathsf{vk}_q$.

If the key $\tilde{\mathsf{vk}}$ used in Σ^* is one of $\mathsf{vk}_1, \ldots, \mathsf{vk}_q$ then $\mathcal{A}$ will immediately lose after phase 1. If $\tilde{\mathsf{vk}}$ is different from all keys in $\mathsf{vk}_1, \ldots, \mathsf{vk}_q$, then $\Sigma^\dagger$ does not link to Σ^*, and consequently $\mathcal{A}$ loses in phase 2. Consequently the advantage of $\mathcal{A}$ in $\mathcal{H}_1$ is 0. This concludes the proof.

7 Conclusions

Ring signatures are a well-studied cryptographic primitive with many applications that include whistleblowing and cryptocurrencies. In this paper we improved the state-of-the-art by introducing a scheme with signature size that is logarithmic in the number of ring members, while at the same time relying on standard assumptions and not requiring a trusted setup. We use novel techniques that combine somewhere statistically binding hashing and NIWI proofs forming a membership proof. An interesting open question is whether one can build structure-preserving SSB hashing that can be combined with pairing based NIWI proofs. Such combination would substantially increase the efficiency of the proposed membership proof and decrease its size.

Acknowledgments. This work has been partially funded/supported by the German Ministry for Education and Research through funding for the project CISPA-Stanford Center for Cybersecurity (Funding numbers: 16KIS0762 and 16KIS0927).

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432, Queenstown, New Zealand, Dec. 1–5, 2002. Springer, Heidelberg, Germany.
2. M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 405–434, Brisbane, Queensland, Australia, Dec. 2–6, 2018. Springer, Heidelberg, Germany.

3. B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Heidelberg, Germany.
4. C. Baum, H. Lin, and S. Oechsner. Towards practical lattice-based one-time linkable ring signatures. Cryptology ePrint Archive, Report 2018/107, 2018. https://eprint.iacr.org/2018/107.
5. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In S. Halevi and T. Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79, New York, NY, USA, Mar. 4–7, 2006. Springer, Heidelberg, Germany.
6. N. Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Y. Kalai and L. Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 567–594, Baltimore, MD, USA, Nov. 12–15, 2017. Springer, Heidelberg, Germany.
7. N. Bitansky and O. Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427, Warsaw, Poland, Mar. 23–25, 2015. Springer, Heidelberg, Germany.
8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.
9. X. Boyen. Mesh signatures. In M. Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 210–227, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.
10. X. Boyen and T. Haines. Forward-secure linkable ring signatures. In W. Susilo and G. Yang, editors, *ACISP 18: 23rd Australasian Conference on Information Security and Privacy*, volume 10946 of *Lecture Notes in Computer Science*, pages 245–264, Wollongong, NSW, Australia, July 11–13, 2018. Springer, Heidelberg, Germany.
11. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, Oct. 22–25, 2011. IEEE Computer Society Press.
12. N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434, Wroclaw, Poland, July 9–13, 2007. Springer, Heidelberg, Germany.
13. S. S. M. Chow, V. K.-W. Wei, J. K. Liu, and T. H. Yuen. Ring signatures without random oracles. In F.-C. Lin, D.-T. Lee, B.-S. Lin, S. Shieh, and S. Jajodia, editors, *ASIACCS 06: 1st ACM Symposium on Information, Computer and Communications Security*, pages 297–302, Taipei, Taiwan, Mar. 21–24, 2006. ACM Press.
14. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

15. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press.
16. C. Dwork and M. Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293, Redondo Beach, CA, USA, Nov. 12–14, 2000. IEEE Computer Society Press.
17. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, Aug. 19–23, 1984. Springer, Heidelberg, Germany.
18. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.
19. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Heidelberg, Germany.
20. E. Ghadafi. Sub-linear blind ring signatures without random oracles. In M. Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 304–323, Oxford, UK, Dec. 17–19, 2013. Springer, Heidelberg, Germany.
21. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press.
22. A. González. A ring signature of size $O(\sqrt[3]{n})$ without random oracles. Cryptology ePrint Archive, Report 2017/905, 2017. http://eprint.iacr.org/2017/905.
23. R. Goyal, S. Hohenberger, V. Koppula, and B. Waters. A generic approach to constructing and proving verifiable random functions. In Y. Kalai and L. Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 537–566, Baltimore, MD, USA, Nov. 12–15, 2017. Springer, Heidelberg, Germany.
24. J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In C. Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, Aug. 20–24, 2006. Springer, Heidelberg, Germany.
25. J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In T. Johansson and S. Maitra, editors, *Progress in Cryptology - INDOCRYPT 2003: 4th International Conference in Cryptology in India*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279, New Delhi, India, Dec. 8–10, 2003. Springer, Heidelberg, Germany.
26. P. Hubacek and D. Wichs. On the communication complexity of secure function evaluation with long output. In T. Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, Jan. 11–13, 2015. Association for Computing Machinery.
27. B. Libert, T. Peters, and C. Qian. Logarithmic-size ring signatures with tight security from the DDH assumption. In J. López, J. Zhou, and M. Soriano, editors, *ESORICS 2018: 23rd European Symposium on Research in Computer Security, Part II*, volume 11099 of *Lecture Notes in Computer Science*, pages 288–308, Barcelona, Spain, Sept. 3–7, 2018. Springer, Heidelberg, Germany.

28. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *ACISP 04: 9th Australasian Conference on Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany.
29. X. Lu, M. H. Au, and Z. Zhang. Raptor: A practical lattice-based (linkable) ring signature. Cryptology ePrint Archive, Report 2018/857, 2018. https://eprint.iacr.org/2018/857.
30. G. Malavolta and D. Schröder. Efficient ring signatures in the standard model. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 128–157, Hong Kong, China, Dec. 3–7, 2017. Springer, Heidelberg, Germany.
31. S. Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. http://eprint.iacr.org/2015/1098.
32. T. Okamoto, K. Pietrzak, B. Waters, and D. Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 121–145, Auckland, New Zealand, Nov. 30 – Dec. 3, 2015. Springer, Heidelberg, Germany.
33. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
34. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, Dec. 9–13, 2001. Springer, Heidelberg, Germany.
35. S. Schäge and J. Schwenk. A CDH-based ring signature scheme with short signatures and public keys. In R. Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 129–142, Tenerife, Canary Islands, Spain, Jan. 25–28, 2010. Springer, Heidelberg, Germany.
36. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, Aug. 20–24, 1990. Springer, Heidelberg, Germany.
37. H. Shacham and B. Waters. Efficient ring signatures without random oracles. In T. Okamoto and X. Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 166–180, Beijing, China, Apr. 16–20, 2007. Springer, Heidelberg, Germany.
38. 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 W. Susilo and G. Yang, editors, *ACISP 18: 23rd Australasian Conference on Information Security and Privacy*, volume 10946 of *Lecture Notes in Computer Science*, pages 558–576, Wollongong, NSW, Australia, July 11–13, 2018. Springer, Heidelberg, Germany.
39. P. P. Tsang and V. K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*, pages 48–60. Springer, 2005.

A Supplementary Material

A.1 Supplementary Preliminaries

We recall the somewhere statistically binding (SSB) hashing definition from [26].

Definition 12. *A somewhere perfectly binding hash family* SPB *is given by a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ *with the following syntax.*

$\mathsf{Gen}(1^\lambda, n, \mathsf{ind})$**:** *Takes as input a security parameter* 1^λ*, a database size* n *and an index* ind *and outputs a hashing key* hk*.*

$\mathsf{Hash}(\mathsf{hk}, \mathsf{db})$**:** *Takes as input a hashing key* hk *and a database* db *and outputs a digest* h*.*

$\mathsf{Open}(\mathsf{hk}, \mathsf{db}, \mathsf{ind})$**:** *Takes as input a hashing key* hk*, a database* db *and an index* ind *and outputs a witness* τ*.*

$\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau)$**:** *Takes as input a hashing key* hk*, a digest* h*, an index* ind*, a value* x *and a witness* τ *and outputs either* 0 *or* 1*.*

To simplify notation, we will not provide the block size of databases as an input to $\mathsf{SPB.Gen}$ *but rather assume that the block size for the specific application context is hardwired in this function.*

We require the following properties.

Correctness: *We say that* $\mathsf{SPB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ *is correct, if it holds for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* db *of size* n *and all indices* $\mathsf{ind} \in [n]$ *that given that* $\mathsf{hk} \leftarrow \mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind})$*,* $\mathsf{h} \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}, \mathsf{db})$ *and* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}, \mathsf{db}, \mathsf{ind})$*, it holds that*

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

Efficiency: *The hashing keys* hk *generated by* $\mathsf{Gen}(1^\lambda, n, \mathsf{ind})$ *and the witnesses* τ *generated by* $\mathsf{Open}(\mathsf{hk}, \mathsf{shk}, \mathsf{db}, \mathsf{ind})$ *are of size* $\log(n) \cdot \mathsf{poly}(\lambda)$*. Moreover,* $\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau)$ *runs in time* $\log(n) \cdot \mathsf{poly}(\lambda)$*.*

Somewhere Perfectly Binding: *It holds for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all random coins* $\mathsf{r}_{\mathsf{Gen}}$*, all databases* db *of size* n*, all indices* $\mathsf{ind} \in [n]$*, all database values* x *and all witnesses* τ *that if* $\mathsf{hk} = \mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind}; \mathsf{r}_{\mathsf{Gen}})$*,* $\mathsf{h} = \mathsf{SPB.Hash}(\mathsf{hk}, \mathsf{db})$ *and* $\mathsf{Verify}(\mathsf{hk}, \mathsf{h}, \mathsf{ind}, x, \tau) = 1$*, then it holds that* $x = \mathsf{db}_{\mathsf{ind}}$*.*

Index Hiding: *Every PPT-adversary* $\mathcal{A}$ *has at most negligible advantage in the following experiment.*

$\mathsf{Exp}(\mathcal{A})$ *:*

1. $\mathcal{A}$ *sends* $(n, \mathsf{ind}_0, \mathsf{ind}_1)$ *to the experiment.*
2. *The experiment chooses a random bit* $b \leftarrow_\$ \{0,1\}$*, computes* $\mathsf{hk} \leftarrow \mathsf{SPB.Gen}(1^\lambda, n, \mathsf{ind}_b)$ *and provides* hk *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a guess* b'*. If* $b' = b$*, the experiment outputs* 1*, otherwise* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathsf{Adv}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}(\mathcal{A}) = 1] - \frac{1}{2}\right|$.

A.2 A Construction of Somewhere Perfectly Binding hashing from the DDH Assumption

In this section, we will recall the DDH-based SSB construction of [32] and show that it fulfils the stronger notion of *Somewhere Perfectly Binding*. Okamoto et al. [32] start by providing a 2-to-1 hash function which perfectly binds to either the left or right input. Combining this construction in a Merkle-tree, where every layer has an associated key, we obtain an SPB hash function with arbitrary compression and local opening.

Let $\mathbb{G}$ be an abelain group of prime order p and let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$. Further assume that the group $\mathbb{G}$ has *constant representation overhead*, that is every element $h \in \mathbb{G}$ can be efficiently encoded by $\log(p) + c = \lambda + c$ bits, where c is a constant. Multiplicative groups of finite fields and certain elliptic curve groups have this property (for more information refer to [32] Section 3).

For a matrix (or vector) $\mathbf{A} \in \mathbb{Z}_p^{m \times n}$ let $g^{\mathbf{A}}$ be the component-wise exponentiation of g with the components of $\mathbf{A}$. Let $d \in \mathbb{N}$. The 2-to-1 SPB scheme, where the Gen algorithm takes the block-length d (encoded unary) as additional input, is given as follows:

$\mathsf{Gen}(1^\lambda, 1^d, \mathsf{ind} \in \{0,1\})$**:**

- Sample $\mathbf{a} \leftarrow (\mathbb{Z}_p \backslash \{0\})^d$, $\mathbf{b} \leftarrow (\mathbb{Z}_p \backslash \{0\})^d$, $\mathbf{w} \leftarrow \mathbb{Z}_p^d$.
- Set $\mathbf{A} \leftarrow \mathbf{w} \cdot \mathbf{a}^\top + (1 - \mathsf{ind}) \cdot \mathbf{I}$ and $\mathbf{B} \leftarrow \mathbf{w} \cdot \mathbf{b}^\top + \mathsf{ind} \cdot \mathbf{I}$.
- Set $\mathbf{g}_0 \leftarrow g^{\mathbf{a}^\top}$, $\mathbf{g}_1 \leftarrow g^{\mathbf{b}^\top}$, $\mathbf{G}_0 \leftarrow g^{\mathbf{A}}$ and $\mathbf{G}_1 \leftarrow g^{\mathbf{B}}$.
- Output $\mathsf{hk} \leftarrow (\mathbf{g}_0, \mathbf{g}_1, \mathbf{G}_0, \mathbf{G}_1)$.

$\mathsf{Hash}(\mathsf{hk} = (\mathbf{h}_0, \mathbf{h}_1, \mathbf{H}_0, \mathbf{H}_1), (\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{Z}_p^d \times \mathbb{Z}_p^d)$**:**

- Compute $h \leftarrow \mathbf{g}_0^{\mathbf{x}_0} \cdot \mathbf{g}_1^{\mathbf{x}_1}$ and $\mathbf{h} \leftarrow \mathbf{G}_0^{\mathbf{x}_0} \cdot \mathbf{G}_1^{\mathbf{x}_1}$.
- Output $\mathsf{h} \leftarrow (h, \mathbf{h})$.

Both the Open and Verify functions are trivial for this construction, as the opening simply consists of $(\mathbf{x}_0, \mathbf{x}_1)$ and verification is performed by hashing $(\mathbf{x}_0, \mathbf{x}_1)$ and testing whether the output is h.

Note that hash values h can be represented by $(d+1) \cdot (\lambda + c)$ bits, whereas each $\mathbf{x}_0$ and $\mathbf{x}_1$ are represented by $d \cdot \lambda$ bits. Thus, encoding hash values requires $\lambda + (d+1) \cdot c$ additional bits compared to $\mathbf{x}_0$ and $\mathbf{x}_1$.

By the decision linear assumption, the key $\mathsf{hk} \leftarrow (\mathbf{g}_0, \mathbf{g}_1, \mathbf{G}_0, \mathbf{G}_1)$ is pseudorandom and therefore hides the index ind, hence the index-hiding property follows immediately.

The somewhere perfectly binding property can be argued as follows. Fix $(\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{Z}_p^d \times \mathbb{Z}_p^d$ and $\mathsf{ind} \in \{0,1\}$. Let $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{ind})$ and $\mathsf{h} \leftarrow \mathsf{Hash}(\mathsf{hk}, (\mathbf{x}_0, \mathbf{x}_1))$. Specifically, we have $\mathsf{hk} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{G}_0, \mathbf{G}_1)$ with $\mathbf{g}_0 = g^{\mathbf{a}^\top}$, $\mathbf{g}_1 = g^{\mathbf{b}^\top}$, $\mathbf{G}_0 = g^{\mathbf{A}} = g^{\mathbf{w} \cdot \mathbf{a}^\top + (1-\mathsf{ind}) \cdot \mathbf{I}}$ and $\mathbf{G}_1 = g^{\mathbf{B}} = g^{\mathbf{w} \cdot \mathbf{b}^\top + \mathsf{ind} \cdot \mathbf{I}}$. Furthermore, it holds that $\mathsf{h} = (h, \mathbf{h})$ with

$$h = \mathbf{g}_0^{\mathbf{x}_0} \cdot \mathbf{g}_1^{\mathbf{x}_1} = g^{\langle \mathbf{a}, \mathbf{x}_0 \rangle + \langle \mathbf{b}, \mathbf{x}_1 \rangle}$$

$$\mathbf{h} = \mathbf{G}_0^{\mathbf{x}_0} \cdot \mathbf{G}_1^{\mathbf{x}_1} = g^{\mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{x}_1} = g^{\mathbf{w} \cdot (\langle \mathbf{a}, \mathbf{x}_0 \rangle + \langle \mathbf{b}, \mathbf{x}_1 \rangle) + (1 - \mathsf{ind}) \cdot \mathbf{x}_0 + \mathsf{ind} \cdot \mathbf{x}_1}.$$

Thus, it holds that

$$\log_g(h) = \langle \mathbf{a}, \mathbf{x}_0 \rangle + \langle \mathbf{b}, \mathbf{x}_1 \rangle$$
$$\log_g(\mathbf{h}) = \mathbf{w} \cdot (\langle \mathbf{a}, \mathbf{x}_0 \rangle + \langle \mathbf{b}, \mathbf{x}_1 \rangle) + (1 - \mathsf{ind}) \cdot \mathbf{x}_0 + \mathsf{ind} \cdot \mathbf{x}_1.$$

Notice that since $\mathbf{a}$ and $\mathbf{b}$ are component-wise non-zero, $\mathbf{w}$ is uniquely determined by $(\mathbf{a}, \mathbf{b}, \mathbf{A}, \mathbf{B})$, and therefore by $\mathsf{hk} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{G}_0, \mathbf{G}_1)$. Consequently, it holds that

$$\log_g(\mathbf{h}) - \mathbf{w} \cdot \log_g(h) = (1 - \mathsf{ind}) \cdot \mathbf{x}_0 + \mathsf{ind} \cdot \mathbf{x}_1 = \mathbf{x}_{\mathsf{ind}},$$

i.e. hk and $\mathsf{h} = (h, \mathbf{h})$ uniquely determine $\mathbf{x}_b$.

SPB Hash with local opening from 2-to-1 SPB Hash. We also briefly recall the Merkle-tree bootstrapping step of [32] to get arbitrary compression and local opening. Recall that the above construction has an overhead in that two blocks, each of size $d \cdot \lambda$ are hashed to a new block of size $d \cdot \lambda + \lambda + (d+1) \cdot c = (d+1) \cdot (1 + c/\lambda) \cdot \lambda$. Thus, when using this 2-to-1 hash function in a Merkle tree, we need to adjust the block size when going up the tree. Specifically, if level 0 is the bottom layer of the tree and we have block-length $d_0 = d$ for the first hash function, we find that block-length $d_i = (d_{i-1} + 1) \cdot (1 + c/\lambda)$ is sufficient for the i-th hash function to be somewhere perfectly binding. Solving this recurrence, we get that

$$\begin{aligned} d_n &= (d_0 - 1 - c/\lambda) \cdot (1 + c/\lambda)^n + (1 + c/\lambda) \cdot \frac{(1 + c/\lambda)^{n+1} - 1}{(1 + c/\lambda) - 1} \\ &\leq d_0 \cdot (1 + c/\lambda)^n + \frac{\lambda}{c} \cdot (1 + c/\lambda)^{n+2} \end{aligned}$$