Repudiable Ring Signature: Stronger Security and Logarithmic-Size

Hao Lin[1,2] , Mingqiang Wang[1,2]

1. School of Mathematics and System Sciences, Shandong University, Jinan, Shandong 250100, PR China;
`lhao17@mail.sdu.edu.cn, wangmingqiang@sdu.edu.cn`
2. China Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education.

Abstract. Ring signatures allow a person to generate a signature on behalf of an ad hoc group, and can hide the true identity of the signer among the group. Repudiable ring signatures are the more strongly defined ring signatures, which can allow every non-signer to prove to others that the signature was not generated by himself.

This paper has two main areas of focus. First, we propose a new requirement for repudiable ring signatures, which is that no one can forge a valid repudiation for others. Second, we present the first logarithmic-size repudiable ring signatures which do not rely on a trusted setup or the random oracle model. Specifically, our scheme can be instantiated from standard assumptions and the size of signatures and repudiations only grows logarithmically in the number of ring members.

Keywords: ring signatures, repudiable ring signatures, logarithmic-size

1 Introduction

Ring signature, introduced by [19], is a variant of digital signature, which can certify that one among a particular set of parties has signed a particular message, without reveal who is the signer. And this particular set is called a 'ring'. 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 produced by one among R. The interesting feature of ring signatures is that given such a signature, no one can tell which key was used to compute this signature. Ring signatures are useful, for example, to certify that certain leaked information comes from a privileged set of government officials without revealing the identity of the whistleblower, to issue important orders or directives without setting up the signer to be a scapegoat for repercussions, or to enable untraceable transactions in cryptocurrencies (such as Monero [16]). In terms of security two properties are required in 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. And anonymity requires that signatures do not give away by which member they were created.

The notion of repudiable ring signatures [18] is an extension of the concept of ring signatures which can allow every non-signer to prove to others that the signature was not generated by himself. More specifically, the repudiable ring signature scheme is a ring signature scheme equipped with an additional pair of algorithms (Repudiate, VerRepud), where Repudiate is an algorithm which can create a repudiation ξ of any signature σ for any non-signer, and VerRepud can verify whether ξ is a valid repudiation.

The repudiability of ring signatures is a necessary property in some situation. For example, we can cite an example from [18] to illustrate the importance of repudiability. Let us consider a hypothetical case, wherein two candidates Alice and Bob are running for president in the land of Oz. Oz is notorious for its petty partisan politics and its tendency to prefer whomever appears friendlier in a series of nationally televised grinning contests between the main-party candidates. At the peak of election season, a disgruntled citizen Eve decides to help out her preferred candidate Bob by publishing the following message, which goes viral on the social networks of Bob supporters:

I created a notorious terrorist group and laundered lots of money!

Signed: Alice or Eve or Alice's campaign chairman.

Of course, the virally publicized message does not actually incriminate Alice at all, since any one of the signatories could have produced it. However, perhaps there is nothing that Alice can do to allay the doubt in the minds of her suspicious detractors.

The reason for this is that ring signatures are deliberately designed to allow anyone to attach anyone else's identity to a signature, without the latter's consent. And just like in the example above, these ring signatures provide protection for malicious people who try to damage the reputation of others. Therefore, we need to use repudiable ring signatures in these situations.

1.1 Our Contributions

In this paper, we focus on both definitions and constructions. We summarize our results in each of these areas, and relate them to prior work.

Definitions of security. Prior work on repudiable ring signature scheme provides definitions of security seem (to us) unnaturally weak, in that they do not address what seem to be valid security concerns. One example is that they did not consider the unforgeability of repudiation. Although at first glance, this property seems to be included in anonymity, it actually requires more in some aspects than anonymity. Because repudiation unforgeability requires that the adversary cannot forge a repudiable of signature σ for others even if he knows the signing key corresponding to the signature σ.

This property is useful in many cases. For example, let us consider the following situation. A company is in trouble and its employees are asked to come up with a solution. Bob is an employee of this company, he thinks of a seemingly feasible solution, but he is afraid of being made a scapegoat by his boss if his solution failed. So he uses a repudiable ring signature to sign his plan, and publishes it. In the end, the plan works and Bob wants to be rewarded by the company alone, so he can forge repudiations for everyone others in his ring. In this case, the ring member can only share the risk but not the reward, which is obviously unfair to the ring members.

The reason for this is that repudiable ring signatures do not satisfy repudiation unforgeability. Therefore, we need repudiation unforgeability in these situations. So in this paper, we formalize the property of repudiation-unforgeability, and give the first construction that satisfies this property.

Constructions. In this paper, we present the first construction of logarithmic-size repudiable ring signatures which do not rely on a trusted setup or the random oracle model. Specifically, our scheme can be instantiated from standard assumptions and our scheme has signatures and repudiations of size $\log(n) \cdot \mathrm{poly}(\lambda)$, whereas the size of the signatures and repudiations of construction in [18] is square in the ring size n.

There are two major obstacles in making the size of the signatures and repudiations sublinear in [18] :

1. The signatures and repudiatons contain all witnesses;
2. The witness for the validity of statement is also size linear in n.

Our first modification is that for signature we can just use NIWI to produce a witness, and do not produce witnesses π for every party, and since NIWI has witness indistinguishable, we also have our signature has anonymity. Our second modification is that we first hash the ring R into a succinct digest h, and then use h in the NIWI. Here we use SPB hashing function, which can also prove the membership of VK_i in the ring R. This hash function was first used by [1].

Besides, the size of keys of our construction has been reduced by at least half compared to scheme in [18].

Other Contributions. We find there are some mistakes in [18]. In [18] they use the notation adaptive anonymity against adversarially chosen keys, but we find their construction cannot satisfy this property they proposed, we can find an attack for their construction. The attack algorithm is in the appendix. To rule out such attack, we need to limit the ability of adversary slightly. Our modification is that, we do not allow an adversary to ask its oracle $\mathrm{OR}(\cdot)$ for $(\cdot, m, \mathrm{R}, \cdot)$ after the adversary gives challenge information $(j_0, j_1, m, \mathrm{R})$ to experiment. Their repudiable ring signatures satisfy this modified anonymity, so do our scheme. And we think this limitation is necessary.

Besides, if we only use the first three algorithms of our repudiable ring signature, it is also a secure ring signature scheme and the size of signatures also grows only logarithmically in the number of ring members. And this is also a new construction of standard ring signatures with logarithmic-size signatures .

1.2 Related Work

After the initial work of Rivest, Shamir and Tauman [19], a number of works provided constructions under various computational hardness assumptions. The scheme of Dodis et al. [6] was the first to achieve sublinear size signatures in the ROM. Libert et al. [12] constructed a scheme with logarithmic size ring signature from DDH in the ROM. And recently, Backes et al. [1] provided a standard model construction with signatures of size $\log(n)$.

And since the original proposal of ring signatures, various variant definitions have been proposed. For example, linkable ring signatures [13] allow identification of signatures that were produced by the same signer, without compromising the anonymity of the signer within the ring. Another notion called traceable ring signature [9] considers a setting where signatures are generated with respect to "tags" and each member may sign at most a single message with respect to a particular tag, or else his identity will be revealed. Accountable ring signatures [4, 20] allow a signer to assign the power to deanonymize his signature to a specific publicly identified party. And recently, Park and Sealfon proposed four new notations which are repudiable, unrepudiable, claimable and unclaimable ring signature in [18].

2 Preliminaries

Throughout the paper, we let λ denote the security parameter and $\mathrm{negl}(\lambda)$ denote the negligible function. We denote by $y \leftarrow \mathcal{A}(x;r)$ the execution of algorithm that $\mathcal{A}$ output y, on input x and random coins r. We write $y \leftarrow \mathcal{A}(x)$, if the specific random coins used are not important. And we denote by $y = \mathcal{A}(x)$, if the algorithm is deterministic. Let $r \leftarrow S$ denote that r is chosen uniformly at random from the set S. We use $[n]$ to denote the set $\{1, \dots, n\}$.

Next, we briefly review some building blocks, which include non-interactive witness-indistinguishable proof system, verifiable random function and somewhere perfectly binding hash function. They will be used in our scheme later.

2.1 Non-Interactive Witness-Indistinguishable Proof System

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

Definition 1 (NIWI). *A non-interactive witness-indistinguishable proof system* NIWI *for language* $\mathcal{L}_{\mathcal{R}}$ *consists of two* PPT *algorithms* (Prove, Verify) *with the following syntax.*

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

Verify (x, π)**:** *takes as input a statement* x *and a proof* π*, outputs either* 0 *or* 1*.*

We require it satisfies the following properties.

Completeness: *for every λ and every $(x, w) \in \mathcal{R}$, we have*

$$\Pr[\text{Verify}(x, \pi) = 1 \mid \pi \leftarrow \text{Prove}(1^\lambda, x, w)] = 1,$$

where the probability is taken over the randomness of Prove *and* Verify *algorithms.*

Soundness: *for every λ, every $x \notin \mathcal{L}_\mathcal{R}$ and every $\pi \in \{0,1\}^*$, we have*

$$\Pr[\text{Verify}(x, \pi) = 1] \leq \text{negl}(\lambda),$$

where the probability is taken over the randomness of Verify *algorithm.*

Witness-Indistinguishability: *for any sequence*

$$\mathcal{I} = \{(x, w_0, w_1) : (x, w_0), (x, w_1) \in \mathcal{R}\},$$

we have

$$\{\text{Prove}(1^\lambda, x, w_0)\}_{(x,w_0,w_1)\in\mathcal{I}} \overset{c}{\approx} \{\text{Prove}(1^\lambda, x, w_1)\}_{(x,w_0,w_1)\in\mathcal{I}}.$$

Just like [1], we also require the size of proof π satisfies $|\pi| = |C_x| \cdot \text{poly}(\lambda)$, where C_x is a verification circuit for the statement x.

NIWI can be constructed from NIZK proofs derandomization assumptions [2, 8], from indistinguishability obfuscation and one-way permutations [3] and from bilinear pairings [10].

To avoid confusion, we will write N.Prove, N.Verify to denote the Prove and Verify algorithms belonging to NIWI.

2.2 Verifiable Random Function

Let $a : \mathbb{N} \to \mathbb{N} \cup \{*\}$ and $b : \mathbb{N} \to \mathbb{N}$ be any two functions such that $a(\lambda)$, $b(\lambda)$ are both computable in time poly(λ), and they are both bounded by a polynomial in λ (expect when $a(\lambda)$ takes on the value $*$).[1]

Definition 2 (VRF). *A verifiable random function* VRF *with input length $a(\lambda)$, output length $b(\lambda)$ consists of a tuple of polynomial-time algorithms* (Gen, Eval, Prove, Verify) *with the following syntax.*

Gen (1^λ)**:** *takes as input a security parameter 1^λ, and outputs a pair of keys (pk, sk), this algorithm is probabilistic.*

Eval (sk, x)**:** *takes as input a secret key sk and $x \in \{0,1\}^{a(\lambda)}$, and outputs $y \in \{0,1\}^{b(\lambda)}$, this algorithm is deterministic.*

Prove (sk, x)**:** *takes as input a secret key sk and $x \in \{0,1\}^{a(\lambda)}$, and outputs a proof π, this algorithm is deterministic.*

Verify (pk, x, y, π)**:** *takes as input a public key pk, $x \in \{0,1\}^{a(\lambda)}$, $y \in \{0,1\}^{b(\lambda)}$, and a proof π, and outputs either* 0 *or* 1*, this algorithm is probabilistic.*

[1] When $a(\lambda)$ takes the value of $*$, it means the VRF is defined for inputs of all length.

We require it satisfies the following properties.

Completeness: *for every λ and every $x \in \{0,1\}^{a(\lambda)}$, we have*

$$\Pr\left[\text{Verify}(pk, x, y, \pi) = 1 \;\middle|\; \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^\lambda); \\ y = \text{Eval}(sk, x); \\ \pi = \text{Prove}(sk, x). \end{array}\right] = 1,$$

where the probability is taken over the randomness of Gen *and* Verify *algorithms.*

Uniqueness: *for every pk, x, y_0, π_0 and y_1, π_1 such that $y_0 \neq y_1$, the following holds for either $i = 0$ or $i = 1$:*

$$\Pr[\text{Verify}(pk, x, y_i, \pi_i) = 1] < \text{negl}(\lambda),$$

where the probability is taken over the randomness of Verify *algorithm.*

Pseudorandomness: *for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have*

$$\Pr\left[b' = b \wedge x \notin \mathcal{Q} \;\middle|\; \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^\lambda); \\ x \leftarrow \mathcal{A}_1^{\text{Eval}(sk,\cdot),\text{Prove}(sk,\cdot)}(1^\lambda, pk); \\ y_0 = \text{Eval}(sk, x), y_1 \leftarrow \{0,1\}^{b(\lambda)}, b \leftarrow \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\text{Eval}(sk,\cdot),\text{Prove}(sk,\cdot)}(1^\lambda, pk, x, y_b). \end{array}\right] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where $\mathcal{Q}$ is the set of oracle queries made by $\mathcal{A}$.

For simplicity, we assume that Eval takes inputs x of any length, i.e. $a(\lambda)$ takes the value of $*$.

The notation of verifiable random function (VRF) was introduced by Micali, Rabin, and Vadhan [15]. Known constructions of VRFs are due to [15] based on strong RSA, [14] based on a strong version of the Diffie-Hellman assumption in bilinear groups, [5] based on the sum-free generalized DDH assumption, and [7] based on the bilinear Diffie-Hellman inversion assumption.

To avoid confusion, we will write V.Gen, V.Eval, V.Prove, V.Verify to denote the Gen, Eval, Prove and Verify algorithms belonging to VRF.

2.3 Somewhere Perfectly Binding Hash Function

The notation of somewhere perfectly binding hash function(SPB)[2] was introduced by [1], which can be used to create a short digest $h = \text{H}_{\text{hk}}(x)$ of some long input $x = (x[1], \ldots, x[L]) \in \Sigma^L$, where Σ is some alphabet. The hashing key $(\text{hk}, \text{shk}) \leftarrow \text{Gen}(i)$ can be generated by providing a special "binding index" i and this ensures that the hash $h = \text{H}_{\text{hk}}(x)$ is perfectly binding for the i'th symbol. In other words, even though h has many other preimages x' such that $\text{H}_{\text{hk}}(x') = h$, all of these preimages agree in the i'th symbol $x'[i] = x[i]$. Moreover, we will be interested in SPB hash function with a 'private local opening' property that allow us to prove that i'th symbol of x takes on some particular value $x[i] = u$ by providing a short opening π. The definition is below.

[2] This is a stronger notion, compare with SSB in [11].

Definition 3 (SPB). *A somewhere perfectly binding hash family with private local opening* SPB *is given by a tuple of algorithms* (Gen, Hash, Open, Verify) *with the following syntax:*

Gen($1^\lambda, n, i$): [3] *Takes as input a security parameter 1^λ, a database size n and an index i, and outputs a hashing key* hk *and a private key* shk.

Hash(hk, x): *Takes as input a hashing key* hk *and a database x and outputs a digest h.*

Open(hk, shk, x, j): *Takes as input a hashing key* hk*, a private key* shk*, a database x and an index j and outputs a witness π.*

Verify(hk, h, j, u, π) *Takes as input a hashing key* hk*, a digest h, an index j, an alphabet u and a witness π, and outputs either* 0 *or* 1*.*

We require the following properties for the SPB*:*

Correctness: *for every security parameter λ, every $n = \mathrm{poly}(\lambda)$, every database x of size n and every index $i \in [n]$, we have*

$$\Pr\left[\mathrm{Verify}(\mathrm{hk}, h, i, x[i], \pi) = 1 \;\middle|\; \begin{array}{l}(\mathrm{hk}, \mathrm{shk}) \leftarrow \mathrm{Gen}(1^\lambda, n, i);\\ h = \mathrm{Hash}(\mathrm{hk}, x);\\ \pi \leftarrow \mathrm{Open}(\mathrm{hk}, \mathrm{shk}, x, i).\end{array}\right] = 1.$$

where the probability is taken over the randomness of Gen*,* Open *and* Verify *algorithm.*

Somewhere Perfectly Binding: *for every security parameter λ, every $n = \mathrm{poly}(\lambda)$, every database x of size n, every index $i \in [n]$, every alphabet value u and every witness π, we have*

$$\Pr\left[i = \mathrm{ind}, u = x[\mathrm{ind}] \;\middle|\; \begin{array}{l}h = \mathrm{Hash}(\mathrm{hk}, x);\\ \mathrm{Verify}(\mathrm{hk}, h, i, u, \pi) = 1.\end{array}\right] = 1.$$

where the hash key hk *is generated by* $\mathrm{Gen}(1^\lambda, n, \mathrm{ind})$.

Index Hiding: *for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l}(n, i_0, i_1) \leftarrow \mathcal{A}_1(1^\lambda);\\ (\mathrm{hk}, \mathrm{shk}) \leftarrow \mathrm{Gen}(1^\lambda, n, i_b), b \leftarrow \{0,1\};\\ b' \leftarrow \mathcal{A}_2(1^\lambda, (\mathrm{hk}, \mathrm{shk})).\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda),$$

Remark 1. We can input any $j \in [n]$ into Open algorithm, but the only j that was used to generate hashing key can produce a valid witness.

Just like [1], we also require the size of hash key hk and the witness π are $\log(n) \cdot \mathrm{poly}(\lambda)$. Moreover, $\mathrm{Verify}(\mathrm{hk}, \mathrm{shk}, i, x, \pi)$ can be computed by a circuit of size $\log(n) \cdot \mathrm{poly}(\lambda)$.

To simplify notation, we will not provide the block size of databases as an input to SPB.Gen but rather assume that the block size for the specific application context is hardwired.

[3] Where we need $i \in [n]$, the same thing has to be true for the following j.

The notation of somewhere perfectly binding hash family with private local opening (SPB) was introduced by [1]. In that work, they give a simple black-box transformation from any SPB hash family to a SPB with private local opening. They also show that the DDH-based SSB construction of [17] can be proofed to be SPB hash family.

To avoid confusion, we will write S.Gen, S.Hash, S.Open, S.Verify to denote the Gen, Hash, Open and Verify algorithms belonging to SPB.

3 Definition of Repudiable Ring Signature

In this section, we provide definitions related to repudiable ring signature, which has a stronger security compared with [18]. Specifically, we require no one can forge a valid repudiation for others, which we call repudiation unforgeability, this is a new requirement we proposed.

Definition 4 (RRS). *A repudiable ring signature scheme is a tuple of* PPT *algorithms* $\mathrm{RRS} = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify}, \mathrm{Repudiate}, \mathrm{VerRepud})$*, satisfying correctness, anonymity, unforgeability, repudiability, repudiation unforgeability. The syntax of* RRS *follows:*

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

Sign(SK, m, R): *takes as input a signing key* SK*, a message* m*, and a set of verification keys* $\mathrm{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$*, and outputs a signature* σ*. The set* R *is also known as a 'ring'.*

Verify(m, R, σ): *takes as input a message* m*, a set of verification keys* $\mathrm{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$*, and a signature* σ*, and outputs either* 0 *or* 1*.*

Repudiate(SK, m, R, σ): *takes as input a signing key* SK*, a message* m*, a set of verification keys* $\mathrm{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$*, and a signature* σ*, and outputs a repudiation* ξ*.*

VerRepud(VK, m, R, σ, ξ): *takes as input a verification key* VK*, a message* m*, a set of verification keys* $\mathrm{R} = (\mathrm{VK}_1, \ldots, \mathrm{VK}_n)$*, a signature* σ*, and a repudiation* ξ*, and outputs either* 0 *or* 1*.*

A repudiable ring signature requires five conditions, expressed by definition 5, 6, 7, 8 and 9 below. Informally, 8 requires that non-signer can repudiate and signer cannot repudiate, 9 requires that no one can forge a valid repudiation for others. To give the formal definition of these properties, we need to introduce three oracles first:

- `Corruption oracle`: For a RRS scheme, the oracle $\mathrm{OC}_{(\mathrm{VK}_1,\mathrm{SK}_1),\ldots,(\mathrm{VK}_l,\mathrm{SK}_l)}$ is defined to take as input $i \in [l]$, and outputs $(\mathrm{VK}_i, \mathrm{SK}_i)$.
- `Signing oracle`: For a RRS scheme, the oracle $\mathrm{OS}_{(\mathrm{VK}_1,\mathrm{SK}_1),\ldots,(\mathrm{VK}_l,\mathrm{SK}_l)}$ is defined to take as input $i \in [l]$, a message m, and a ring R[4], and output $\mathrm{Sign}(\mathrm{SK}_i, m, \mathrm{R})$.

[4] We allow that the ring R may contain maliciously chosen verification keys that were not included in $\{\mathrm{VK}_1, \ldots, \mathrm{VK}_l\}$. The same thing holds for the ring in the OR.

– `Repudiation oracle`: For a RRS scheme, the oracle $\mathrm{OR}_{(\mathrm{VK}_1,\mathrm{SK}_1),\ldots,(\mathrm{VK}_l,\mathrm{SK}_l)}$ is defined to take as input $j \in [l]$, a message m, a ring R, and a signature σ, and output $\mathrm{Repudiate}(\mathrm{SK}_j, m, \mathrm{R}, \sigma)$.

Definition 5 (Correctness). *We say that a* RRS *scheme satisfies correctness, if for every security parameter λ, every $n = \mathrm{poly}(\lambda)$, every $j \in [n]$, every message m, we have*

$$\Pr\left[\mathrm{Verify}(m, \mathrm{R}, \sigma) = 1 \;\middle|\; \begin{array}{l} (\mathrm{VK}_{i_1}, \mathrm{SK}_{i_1}), \ldots, (\mathrm{VK}_{i_n}, \mathrm{SK}_{i_n}) \leftarrow \mathrm{Gen}(1^\lambda); \\ \mathrm{R} = \{\mathrm{VK}_{i_1}, \ldots, \mathrm{VK}_{i_n}\}; \\ \sigma = \mathrm{Sign}(\mathrm{SK}_{i_j}, m, \mathrm{R}). \end{array}\right] = 1.$$

where the probability is taken over the randomness of Gen*,* Sign *and* Verify *algorithm.*

In this paper, we refer to adaptive anonymity against adversarially chosen keys, which is slightly different from the previous one in [18]. We require the adversary cannot queries its $\mathrm{OC}(\cdot)$ oracle at $(\cdot, m, \mathrm{R}, \cdot)$, where m and R are the challenge message and ring in the following experiment. In fact, the RRS scheme in [18] only satisfies our anonymity definition and do not satisfies the definition they proposed.

Definition 6 (Anonymity). *We say that a* RRS *scheme satisfies adaptive anonymity against adversarially chosen keys, if for every security parameter λ, every* PPT *adversary $\mathcal{A}$, every $l = \mathrm{poly}(\lambda)$, it holds that $\mathcal{A}$ has at most negligible advantage in the following experiment.*

$\mathrm{Exp}_{\mathrm{Ano}}(\mathcal{A})$*:*

1. *For all $i \in [l]$, the experiment generates the key pairs $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$.*
2. *$\mathcal{A}$ is given input 1^λ, $\mathrm{VK}_1, \ldots, \mathrm{VK}_l$, and oracle access to $\mathrm{OC}(\cdot)$, $\mathrm{OS}(\cdot)$, $\mathrm{OR}(\cdot)$, and then provides a tuple $(m, \mathrm{R}, j_0, j_1)$ to experiment, with $j_0, j_1 \in [l]$, and $\mathrm{VK}_{j_0}, \mathrm{VK}_{j_1} \in \mathrm{R}$.*
3. *The experiment chooses a random bit $b \leftarrow \{0, 1\}$, and computes $\sigma \leftarrow \mathrm{Sign}(\mathrm{SK}_{j_b}, m, \mathrm{R})$, then gives σ to $\mathcal{A}$.*
4. *$\mathcal{A}$ continues to have oracle access to $\mathrm{OC}(\cdot)$, $\mathrm{OS}(\cdot)$ and $\mathrm{OR}(\cdot)$ except that $\mathcal{A}$ can not query $\mathrm{OR}(\cdot)$ with $(\cdot, m, \mathrm{R}, \cdot)$. Finally $\mathcal{A}$ outputs a guess b'. Let $\mathcal{Q}_{\mathrm{OC}}$ denote the set of all queries that $\mathcal{A}$ asked its oracle* OC*.*
5. *The experiment outputs 1, if $j_0, j_1 \notin \mathcal{Q}_{\mathrm{OC}}$, otherwise, outputs 0.*

The advantage of $\mathcal{A}$ is defined by $\mathrm{Adv}_{\mathrm{Ano}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\mathrm{Ano}}(\mathcal{A}) = 1] - \frac{1}{2}$.

Remark 2. We allow that ring R chosen by adversary $\mathcal{A}$ in step 2 may contain maliciously chosen verification keys that were not generated by challenger.

Definition 7 (Unforgeability). *We say that a* RRS *scheme is unforgeable with respect to insider corruption, if for every security parameter λ, every* PPT *adversary $\mathcal{A}$, every $l = \mathrm{poly}(\lambda)$, it holds that $\mathcal{A}$ has at most negligible advantage in the following experiment.*

$\mathrm{Exp}_{\mathrm{Unf}}(\mathcal{A})$*:*

1. *For all* $i \in [l]$*, the experiment generates the key pairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$.
2. $\mathcal{A}$ *is given input* 1^λ, $\mathrm{VK}_1, \dots, \mathrm{VK}_l$*, and oracle access to* $\mathrm{OC}(\cdot)$, $\mathrm{OS}(\cdot)$, $\mathrm{OR}(\cdot)$*, and then outputs* (m, R, σ)*. Let* $\mathcal{Q}_{\mathrm{OC}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OC*, and* $\mathcal{Q}_{\mathrm{OS}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OS.
3. *The experiment outputs* 1*, if it satisfies that* $\mathrm{Verify}(m, \mathrm{R}, \sigma) = 1$, $\mathrm{R} \subset \{\mathrm{VK}_1, \dots, \mathrm{VK}_l\} \setminus \mathcal{Q}_{\mathrm{OC}}$, $(\cdot, m, \mathrm{R}) \notin \mathcal{Q}_{\mathrm{OS}}$*. Otherwise, the experiment outputs* 0.

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

Repudiability requires two condition, which expressed by the following two experiments $\mathrm{Exp}_{\mathrm{Rep1}}$ and $\mathrm{Exp}_{\mathrm{Rep2}}$. Intuitively, $\mathrm{Exp}_{\mathrm{Rep1}}$ captures the requirement that for any signature, a non signer can produce a valid repudiation; $\mathrm{Exp}_{\mathrm{Rep2}}$ captures the requirement that the signer cannot produce a valid repudaition. This requirement was proposed by [18] first.

Definition 8 (Repudiability). *We say that a* RRS *scheme is repudiable with respect to insider corruption, if for every security parameter* λ*, every* PPT *adversary* $\mathcal{A}$*, every* $l = \mathrm{poly}(\lambda)$*, it holds that* $\mathcal{A}$ *has at most negligible advantage in the following two experiments.*

$\mathrm{Exp}_{\mathrm{Rep1}}(\mathcal{A})$ *(Non-signer can repudiate):*

1. *For all* $i \in [l]$*, the experiment generates the key pairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$.
2. $\mathcal{A}$ *is given input* 1^λ, $\mathrm{VK}_1, \dots, \mathrm{VK}_l$*, and oracle access to* $\mathrm{OC}(\cdot)$, $\mathrm{OS}(\cdot)$, $\mathrm{OR}(\cdot)$*. Then* $\mathcal{A}$ *outputs* (m, R, σ) *with* $\mathrm{R} \subset \{\mathrm{VK}_1, \dots, \mathrm{VK}_l\}$*. Let* $\mathcal{Q}_{\mathrm{OC}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OC, $\mathcal{Q}_{\mathrm{OS}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OS*, and* $\mathcal{Q}_{\mathrm{OR}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OR.
3. *For all* $\mathrm{VK}_j \in \mathrm{R} \backslash \mathcal{Q}_{\mathrm{OC}}$*, the experiment computes* $\xi_j = \mathrm{Repudiate}(\mathrm{SK}_j, m, \mathrm{R}, \sigma)$*, and* $b_j = \mathrm{VerRepud}(\mathrm{VK}_j, m, \mathrm{R}, \sigma, \xi_j)$.
4. *The experiment outputs* 1*, if it satisfies that* $(\cdot, m, \mathrm{R}) \notin \mathcal{Q}_{\mathrm{OS}}$, $(\cdot, m, \mathrm{R}, \cdot) \notin \mathcal{Q}_{\mathrm{OR}}$, $\mathrm{Verify}(m, \mathrm{R}, \sigma) = 1$, $\mathrm{R} \setminus \mathcal{Q}_{\mathrm{OC}} \neq \emptyset$, $\bigwedge_{\mathrm{VK}_j \in \mathrm{R} \backslash \mathcal{Q}_{\mathrm{OC}}} b_j = 0$*. Otherwise, the experiment outputs* 0.

The advantage of $\mathcal{A}$ *is defined by* $\mathrm{Adv}_{\mathrm{Rep1}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\mathrm{Rep1}}(\mathcal{A}) = 1]$.

$\mathrm{Exp}_{\mathrm{Rep2}}(\mathcal{A})$*(Signer cannot repudiate):*

1. *For all* $i \in [l]$*, the experiment generates the key pairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$.
2. $\mathcal{A}$ *is given input* 1^λ, $\mathrm{VK}_1, \dots, \mathrm{VK}_l$*, and oracle access to* $\mathrm{OC}(\cdot)$, $\mathrm{OS}(\cdot)$, $\mathrm{OR}(\cdot)$*. Then* $\mathcal{A}$ *outputs* $(m, \mathrm{R}, \sigma, \{\xi_{i_k}\}_{\mathrm{VK}_{i_k} \in \mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R}})$ *with* $\mathrm{R} \subset \{\mathrm{VK}_1, \dots, \mathrm{VK}_l\}$*. Where* $\mathcal{Q}_{\mathrm{OC}}$ *is the set of all queries that* $\mathcal{A}$ *asked its oracle* OC*, and let* $\mathcal{Q}_{\mathrm{OS}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OS.
3. *The experiment compute* $b_{i_k} = \mathrm{VerRepud}(\mathrm{VK}_{i_k}, m, \mathrm{R}, \sigma, \xi_{i_k})$ *for all* $\mathrm{VK}_{i_k} \in \mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R}$.

4. *The experiment outputs* 1*, if it satisfies that* $(\cdot, m, \mathrm{R}) \notin \mathcal{Q}_{\mathrm{OS}}$*,* $\mathrm{Verify}(m, \mathrm{R}, \sigma) = 1$*,* $\mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R} \neq \emptyset$*,* $\bigwedge_{\mathrm{VK}_j \in \mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R}} b_j = 1$*. Otherwise, the experiment outputs* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathrm{Adv}_{\mathrm{Rep2}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\mathrm{Rep2}}(\mathcal{A}) = 1]$*.*

Furthermore, we also need that no one can produce a valid repudiation for others, even if he is signer. We call this requirement repudiation unforgeability, and this is a new requirement we proposed. Repudiation unforgeability requires two conditions, which expressed by the following two experiment $\mathrm{Exp}_{\mathrm{Reun1}}$ and $\mathrm{Exp}_{\mathrm{Reun2}}$. Intuitively, $\mathrm{Exp}_{\mathrm{Reun1}}$ captures the requirement that signer cannot forge repudiation for others; $\mathrm{Exp}_{\mathrm{Reun2}}$ captures the requirement that non-signer also cannot forge repudiation for others.

Definition 9 (Repudiation-Unforgeability). *We say that a* RRS *scheme satisfies repudiation unforgeability, if for every security parameter* λ*, every* PPT *adversary* $\mathcal{A}$*, every* $l = \mathrm{poly}(\lambda)$*, it holds that* $\mathcal{A}$ *has at most negligible advantage in the following two experiments.*

$\mathrm{Exp}_{\mathrm{Reun1}}(\mathcal{A})$ *(Signer cannot forge repudiation for others):*

1. *For all* $i \in [l]$*, the experiment generates the key pairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$*.*
2. $\mathcal{A}$ *is given input* 1^λ*,* $\mathrm{VK}_1, \ldots, \mathrm{VK}_l$*, and oracle access to* $\mathrm{OC}(\cdot)$*,* $\mathrm{OS}(\cdot)$*,* $\mathrm{OR}(\cdot)$*. Then* $\mathcal{A}$ *outputs* $(m, \mathrm{R}, \sigma, \xi, j)$ *with* $\mathrm{R} \subset \{\mathrm{VK}_1, \cdots, \mathrm{VK}_l\}$*,* $\mathrm{VK}_j \in \mathrm{R}$*. Let* $\mathcal{Q}_{OC}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OC*,* $\mathcal{Q}_{OS}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OS*, and* $\mathcal{Q}_{OR}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OR*.*
3. *The experiment outputs* 1*, if it satisfies that* $(\cdot, m, \mathrm{R}) \notin \mathcal{Q}_{\mathrm{OS}}$*,* $\mathrm{Verify}(m, \mathrm{R}, \sigma) = 1$*,* $j \notin \mathcal{Q}_{\mathrm{OC}}$*,* $(j, m, \mathrm{R}, \cdot) \notin \mathcal{Q}_{\mathrm{OR}}$*,* $\mathrm{VerRepud}(\mathrm{VK}_j, m, \mathrm{R}, \sigma, \xi) = 1$*. Otherwise, the experiment outputs* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathrm{Adv}_{\mathrm{Reun2}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\mathrm{Reun1}}(\mathcal{A}) = 1]$*.*

$\mathrm{Exp}_{\mathrm{Reun2}}(\mathcal{A})$ *(Non-signer cannot forge repudiation for others):*

1. *For all* $i \in [l]$*, the experiment generates the key pairs* $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathrm{Gen}(\lambda)$*.*
2. $\mathcal{A}$ *is given input* 1^λ*,* $\mathrm{VK}_1, \ldots, \mathrm{VK}_l$*, and oracle access to* $\mathrm{OC}(\cdot)$*,* $\mathrm{OS}(\cdot)$*,* $\mathrm{OR}(\cdot)$*. Then* $\mathcal{A}$ *outputs a tuple* (m, R, i) *to experiment, with* $i \in [l]$*, and* $\mathrm{VK}_i \in \mathrm{R}$*.*
3. *The experiment computes* $\sigma \leftarrow \mathrm{Sign}(\mathrm{SK}_i, m, \mathrm{R})$*, then give* σ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *continues to have oracle access to* $\mathrm{OC}(\cdot)$*,* $\mathrm{OS}(\cdot)$*. Then* $\mathcal{A}$ *outputs* (j, ξ)*. Let* $\mathcal{Q}_{\mathrm{OC}}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle* OC*.*
5. *The experiment outputs* 1*, if it satisfies that* $\mathrm{VerRepud}(\mathrm{VK}_j, m, \mathrm{R}, \sigma, \xi) = 1$*,* $j \notin \mathcal{Q}_{\mathrm{OC}}$*. Otherwise, the experiment outputs* 0*.*

The advantage of $\mathcal{A}$ *is defined by* $\mathrm{Adv}_{\mathrm{Reun2}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\mathrm{Reun2}}(\mathcal{A}) = 1]$*.*

Remark 3. In the experiment $\mathrm{Exp}_{\mathrm{Reun2}}$, we consider a stronger requirement that the adversary can even know the signing key of signature , i.e. $i \in \mathcal{Q}_{\mathrm{OC}}$they still cannot forge a valid repudiation.

4 Construction of Repudiable Ring Signatures

We now describe our construction of a repudiable ring signature scheme that satisfies a stronger of our proposed definitions, has logarithmic-size signature and reputation, and is based on general assumptions. Let (S.Gen, S.Hash, S.Open, S.Verify) be a somewhere perfectly binding hash function with private local opening, let (V.Gen, V.Eval, V.Prove, V.Verify) be a verifiable random function with input domain $\{0,1\}^*$, output range $\{0,1\}^{\alpha(\lambda)}$, and let (N.Prove, N.Verify) be a NIWI-proof system for an NP-language $\mathcal{L}$ (that will become clear once we describe the scheme). In the rest of the section, we use the following convention to parse a ring R, write $\mathrm{R} = \{\mathrm{VK}_{i_1}, \dots, \mathrm{VK}_{i_n}\}$, and we use R$[k]$ denote k'th verification key VK_{i_k} in R.

The idea underlying our construction is the following. Each user has two VRF key pairs (pk^0, sk^0), (pk^1, sk^1). To generate a ring signature with a signing key SK_{i_j}, the signer first generates two hashing keys hk_0 and hk_1 which are binding at position j and computes the hash of $\mathrm{R} = \{\mathrm{VK}_{i_1}, \dots, \mathrm{VK}_{i_n}\}$ under both hk_0 and hk_1, obtaining hash values h_0 and h_1. Next, the signer computes evaluations y_0^0 and y_0^1 on $(h_0, m; r)$ using sk^0 and sk^1 separately. The evaluations y_1^0 and y_1^1 are chosen uniformly random. Finally, the signer computes a NIWI proof π which proves that either (hk_0, h_0) bind to a key VK_{i_j} and that y_0^0 and y_0^1 are evaluations of $(h_0, m; r)$ for sk^0 and sk^1 or (hk_1, h_1) bind to a key VK_{i_j} and that y_1^0 and y_1^1 are evaluations of $(h_0, m; r)$ for sk^0 and sk^1.

To generate a repudiation for a signature σ with a signing key SK_{i_j}, the denier first computes the hash of $\mathrm{R} = \{\mathrm{VK}_{i_1}, \dots, \mathrm{VK}_{i_n}\}$ under both hk_0 and hk_1, obtaining hash values $\tilde{h}_0$ and $\tilde{h}_1$. Next, the denier computes evaluations y_0 and y_1 on $(\tilde{h}_0, m; r)$ and $(\tilde{h}_1, m; r)$ using sk^1 separately, and then computes evaluations z_0 and z_1 on y_0 and y_1 using sk^1 separately. Finally, the denier computes a NIWI proof π' which proves that z_0 and z_1 are evaluations of y_0 and y_1 for sk^1, and y_0 and y_1 are evaluations of $(\tilde{h}_0, m; r)$ and $(\tilde{h}_1, m; r)$ for sk^1, and $y_0 \neq y_0^1$, $y_1 \neq y_1^1$, where y_0^1 and y_1^1 come from σ.

Toward a formal description, let $\mathcal{L}_1$ denote the NP language:

$$(m, r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, h_0, h_1) \in \mathcal{L}_1 \Leftrightarrow$$

$$\exists\ \mathrm{VK}, i, \eta, \tau^0, \tau^1, j \in \{0,1\} \quad s.t.\ \mathrm{S.Verify}(\mathrm{hk}_j, h_j, i, \mathrm{VK}, \eta) = 1 \wedge$$

$$\mathrm{V.Verify}(pk^0, (h_j, m; r), y_j^0, \tau^0) = 1 \wedge \mathrm{V.Verify}(pk^1, (h_j, m; r), y_j^1, \tau^1) = 1.$$

i.e. $(m, \varphi, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, h_0, h_1) \in \mathcal{L}_1$ iff either (hk_0, h_0) bind to a key VK, y_0^0 and y_0^1 are evaluations of $(h_0, m; r)$ for sk^0 and sk^1 or (hk_1, h_1) bind to a key VK, y_1^0 and y_1^1 are evaluations of $(h_0, m; r)$ for sk^0 and sk^1. This NP language is used to produce ring signature.

And we let $\mathcal{L}_2$ denote the NP language:

$$(\mathrm{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0, \tilde{h}_1, z_0, z_1) \in \mathcal{L}_2 \Leftrightarrow$$

$$\exists y_0, y_1, \tau_{00}, \tau_{01}, \tau_{10}, \tau_{11} \quad s.t.\ (y_0 \neq y_0^1) \wedge (y_1 \neq y_1^1) \wedge$$

$$\text{V.Verify}(pk^1, (\tilde{h}_0, m; r), y_0, \tau_{00}) = 1 \wedge \text{V.Verify}(pk^1, y_0, z_0, \tau_{01}) = 1 \wedge$$

$$\text{V.Verify}(pk^1, (\tilde{h}_1, m; r), y_1, \tau_{10}) = 1 \wedge \text{V.Verify}(pk^1, y_1, z_1, \tau_{11}) = 1.$$

i.e. $(\text{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0, \tilde{h}_1, z_0, z_1) \in \mathcal{L}_2$ iff z_0 and z_1 are evaluations of y_0 and y_1 for sk^1, where y_0 and y_1 are evaluations of $(\tilde{h}_0, m; r)$ and $(\tilde{h}_1, m; r)$ for sk^1, and $y_0 \neq y_0^1$, $y_1 \neq y_1^1$. This NP language is used to produce repudiation.

4.1 Construction

Our repudiable ring signature schemes RRS = (Gen, Sign, Verify, Repudiate, VerRepud) is given as follows.

– RRS.Gen(1^λ):
1. Generate VRF key pairs (pk^0, sk^0), $(pk^1, sk^1) \leftarrow \text{V.Gen}(1^\lambda)$.
2. Output the public key $\text{VK} = (pk^0, pk^1)$, and the secret key $\text{SK} = (sk^0, sk^1, \text{VK})$.[5]

– RRS.Sign(SK, m, R):
1. Parse R as described above and SK=$(sk^0, sk^1$, VK), if VK $\notin$ R, output $\perp$ and halt, else define $i^* \in [N]$ such that $\text{R}[i^*] = \text{VK}$.
2. Generate SPB key pairs $(\text{hk}_0, \text{shk}_0)$, $(\text{hk}_1, \text{shk}_1) \leftarrow \text{S.Gen}(1^\lambda, |\text{R}|, i^*)$, and compute $h_0 = \text{S.Hash}(\text{hk}_0, \text{R})$, $h_1 = \text{S.Hash}(\text{hk}_1, \text{R})$, and then compute $\eta \leftarrow \text{S.Open}(\text{hk}_0, \text{shk}_0, \text{R}, i^*)$.
3. Choose random bit string $r \leftarrow \{0,1\}^\lambda$, and compute $y_0^0 = \text{V.Eval}(sk^0, (h_0, m; r))$, $y_0^1 = \text{V.Eval}(sk^1, (h_0, m; r))$, and then compute proofs $\tau^0 = \text{V.Prove}(sk^0, (h_0, m; r))$, $\tau^1 = \text{V.Prove}(sk^1, (h_0, m; r))$.
4. Choose random bit strings $y_1^0 \leftarrow \{0,1\}^{\alpha(\lambda)}$, $y_1^1 \leftarrow \{0,1\}^{\alpha(\lambda)}$.
5. Set $x = (m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1)$ and $w = (\text{VK}, i^*, \eta, \tau^0, \tau^1, 0)$, and then compute the proof $\pi \leftarrow \text{N.Prove}_{\mathcal{L}_1}(x, w)$.
6. The ring signature is $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$.

– RRS.Verify(m, R, σ):
1. Parse R as above and $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$.
2. Compute $h_0' = \text{S.Hash}(\text{hk}_0, \text{R})$, $h_1' = \text{S.Hash}(\text{hk}_1, \text{R})$.
3. Output $\text{N.Verify}_{\mathcal{L}_1}((m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0', h_1'), \pi)$.

The above is our ring signature algorithm, this algorithm can be used as a general ring signature system alone. Now we proceed to describe the repudiation algorithms for RRS.

– RRS.Repudiate(SK, m, R, σ):
1. Parse R as above, SK= $(sk^0, sk^1$, VK), and $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$, if VK $\notin$ R, output $\perp$ and halt. Else compute $b = \text{RRS.Verify}(m, \text{R}, \sigma)$, if $b = 0$ output $\perp$ and halt.
2. Compute $\tilde{h}_0 = \text{S.Hash}(\text{hk}_0, \text{R})$, $\tilde{h}_1 = \text{S.Hash}(\text{hk}_1, \text{R})$.

[5] We include the verification key VK in SK so that the Sign procedure can identify the verification key in the ring corresponding to the signing key.

3. Compute $y_0 = \text{V.Eval}(sk^1, (\tilde{h}_0, m; r))$ and $y_1 = \text{V.Eval}(sk^1, (\tilde{h}_1, m; r))$. If $y_0 = y_0^1$ or $y_1 = y_1^1$, output $\perp$ and halt.
4. Compute $z_0 = \text{V.Eval}(sk^1, y_0)$, $z_1 = \text{V.Eval}(sk^1, y_1)$, and then compute proofs $\tau_{00} = \text{V.Prove}(sk^1, (\tilde{h}_0, m; r))$, $\tau_{10} = \text{V.Prove}(sk^1, (\tilde{h}_1, m; r))$, and $\tau_{01} = \text{V.Prove}(sk^1, y_0)$, $\tau_{11} = \text{V.Prove}(sk^1, y_1)$.
5. Set $x = (\text{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0, \tilde{h}_1, z_0, z_1)$ and $w = (y_0, y_1, \tau_{00}, \tau_{01}, \tau_{10}, \tau_{11})$, and then compute the proof $\pi' \leftarrow \text{N.Prove}_{\mathcal{L}_2}(x, w)$.
6. The reputation is $\xi = (z_0, z_1, \pi')$.

– RRS.VerRepud(VK, m, R, σ, ξ):
1. Parse R as above, $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$, and $\xi = (z_0, z_1, \pi')$.
2. If VK $\notin$ R, output 1 and halt. Compute $b = \text{RRS.Verify}(m, \text{R}, \sigma)$, if $b = 0$ output 1 and halt.
3. Compute $\tilde{h}_0' = \text{S.Hash}(\text{hk}_0, \text{R})$, $\tilde{h}_1' = \text{S.Hash}(\text{hk}_1, \text{R})$.
4. Output $\text{N.Verify}_{\mathcal{L}_2}((\text{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0', \tilde{h}_1', z_0, z_1), \pi')$.

4.2 Signature and Repudiation Size

We will first show that our scheme has only logarithmic-size signatures and repudiations. For a signature $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$, the size of r, y_0^0, y_0^1, y_1^0, y_1^1 is poly(λ) and independent of the ring-size n. Since SPB is efficient, we have hk_0, hk_1 is bounded by $\log(n) \cdot \text{poly}(\lambda)$. Also by the efficiency of SPB the size of the witness τ is $\log(n) \cdot \text{poly}(\lambda)$ and the SPB verification function S.Verify can be computed by a circuit of size $\log(n) \cdot \text{poly}(\lambda)$. Therefore, the verification circuit C_x for the language $\mathcal{L}_1$ and statement $x = (m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1)$ has size $\log(n) \cdot \text{poly}(\lambda)$. By the proof-size property of the NIWI proof it holds that $|\pi| = |C_x| \cdot \text{poly}(\lambda) = \log(n) \cdot \text{poly}(\lambda)$. Consequently, the size of signatures σ is $\log(n) \cdot \text{poly}(\lambda)$.

For a repudiation $\xi = (z_0, z_1, \pi')$, the size of z_0, z_1 is poly(λ) and independent of the ring-size n. Using the same analysis we can also get that the size of proof π' is $\log(n) \cdot \text{poly}(\lambda)$. Consequently, the size of repudiations ξ is $\log(n) \cdot \text{poly}(\lambda)$.

4.3 Correctness

We now show that our scheme satisfies correctness.

Theorem 1. *The repudiable ring signature scheme* RRS *is correct, given that* NIWI *has completeness,* VRF *has completeness, and* SPB *has correctness.*

Proof. Assume that (VK, SK) were generated by RRS.Gen(1^λ), and signature $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$ is the output of $\text{RRS.Sign}(\text{SK}, m, \text{R})$, where $\text{R} = (\text{VK}_1, \ldots \text{VK}_n)$ is a ring generated by RRS.Gen(1^λ), and R[i] = VK. We will show that it holds $\text{RRS.Verify}(m, \text{R}, \sigma) = 1$. First note that since S.Hash is a deterministic algorithm, it holds $h_0' = h_0$ and $h_1' = h_1$. According to the RRS.Sign algorithm and the correctness of SPB, we have there is a η, such that it holds that $\text{S.Verify}(\text{hk}_0, h_0, i, \text{VK}, \eta) = 1$. Moreover, by the completeness of

VRF, there are $\tau^0,\ \tau^1$ such that it holds $\text{V.Verify}(pk^0, (h_0, m; r), y_0^0, \tau^0) = 1$ and $\text{V.Verify}(pk^1, (h_0, m; r), y_0^1, \tau^1) = 1$.

Therefore, $(m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1) \in \mathcal{L}_1$ and $(\text{VK}, i, \eta, \tau^0, \tau^1, 0)$ is a witness for the membership. Thus, by the correctness of NIWI it holds that

$$\text{N.Verify}_{\mathcal{L}_1}((m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1), \pi) = 1.$$

and consequently $\text{RRS.Verify}(m, \text{R}, \sigma) = 1$.

Remark 4. Definition 5 considers only for honestly generated keys. We can also consider a stronger requirement that verify be successful for honestly generated signatures with respect to rings containing adversarial keys. And we can easily proof that our construction also satisfy this stronger requirement by the same way.

4.4 Repudiation unforgeability

We will turn to show that our RRS scheme is repudiation unforgeability.

Theorem 2. *The repudiable ring signature scheme* RRS *satisfies repudiation unforgeability, given that* NIWI *has soundness,* VRF *has completeness, uniqueness, and pseudorandomness.*

The main idea of the proof is that, if $\mathcal{A}$ can forge a valid repudiation for other member, then by the soundness of NIWI, there must be y_0, τ_{00}, τ_{01}, s.t. $\text{V.Verify}(pk^1, (h_0, m; r), y_0, \tau_{00}) = 1$ and $\text{V.Verify}(pk^1, y_0, z_0, \tau_{01}) = 1$. Since the VRF has completeness and uniqueness, we have $y_0 = \text{V.Eval}(sk^1, (h_0, m; r))$, and $z_0 = \text{V.Eval}(sk^1, y_0)$, and by this we can attack the pseudorandomness of VRF.

Proof. We will prove each of the desired security properties in turn.

Signer cannot forge repudiations for others: Assume there exists a PPT adversary $\mathcal{A}$ that breaks our RRS scheme (in the sense of $\text{Exp}_{\text{Reun1}}$) with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability. The reduction is given as follows.

Adversary $\mathcal{B}$: given input 1^λ, pk, and access to oracle $\text{Eval}(sk, \cdot)$, $\text{Prove}(sk, \cdot)$.

1. $\mathcal{B}$ runs $\text{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\text{VK}_{i^*} = (pk_{i^*}^0, pk)$. Then $\mathcal{B}$ gives 1^λ and $\text{VK}_1, \ldots, \text{VK}_l$ to $\mathcal{A}$.
2. $\mathcal{B}$ proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:
   – If $\mathcal{A}$ queries its corruption oracle $\text{OC}(\cdot)$ on a user $i \neq i^*$, $\mathcal{B}$ faithfully answers SK_i to $\mathcal{A}$. If $\mathcal{A}$ makes a corruption query for i^*, then $\mathcal{B}$ simply aborts.

  - If $\mathcal{A}$ queries its signing oracle $\mathrm{OS}(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Sign}(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathcal{A}$. If $\mathcal{A}$ makes a signing query for (i^*, m, R), then $\mathcal{B}$ runs the honest signing algorithm RRS.Sign with the following modification: in step 3, instead of using sk to generate y_0^1 and τ^1, $\mathcal{B}$ generates these by invoking its VRF oracle.
  - If $\mathcal{A}$ queries its repudiation oracle $\mathrm{OR}(\cdot)$ on $(i, m, \mathrm{R}, \sigma)$, where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Repudiate}(\mathrm{SK}_i, m, \mathrm{R}, \sigma)$ to $\mathcal{A}$. If $\mathcal{A}$ makes a repudiation query for $(i^*, m, \mathrm{R}, \sigma)$, then $\mathcal{B}$ runs the honest repudiating algorithm RRS.Repudiate with the following modification: in step 3, step 4, instead of using sk to generate y_0, y_1, τ_{00}, τ_{10}, z_0, z_1, and τ_{01}, τ_{11}, $\mathcal{B}$ generates these by invoking its VRF oracle.

3. Finally $\mathcal{A}$ outputs $(m, \mathrm{R}, \sigma, \xi, j)$. If $j \neq i^*$, then $\mathcal{B}$ simply aborts. If $j = i^*$, then $\mathcal{B}$ parse $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$ and $\xi = (z_0, z_1, \pi')$, and do:
   - Compute $h_0 = \mathrm{SPB.Hash}(\mathrm{hk}_0, \mathrm{R})$;
   - $\mathcal{B}$ queries its oracle $\mathrm{Eval}(\cdot)$ on $(h_0, m; r)$ and get x.
   - $\mathcal{B}$ submits x to the VRF challenger and then receives responses y. If $y = z_0$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

It remains to show that the adversary $\mathcal{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. First note that the distribution (i.e., verification keys and oracle responses) of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathcal{A}$ does not trigger this abort. And, conditioned that $\mathcal{A}$ does not ask to corrupt i^*, from the view of $\mathcal{A}$ the index i^* is distributed uniformly random, so it holds that $j = i^*$ with probability at least $\frac{1}{l}$.

Now, we assume that no abort happened and $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Reun1}}$. Then ξ will be a valid repudiation of i^* respect to σ, i.e.

$$\mathrm{RRS.VerRepud}(\mathrm{VK}_{i^*}, m, \mathrm{R}, \sigma, \xi) = 1.$$

Therefore, by the soundness of NIWI, there must exist y_0, and τ_{00}, τ_{01}, s.t. we have $\mathrm{V.Verify}(pk, (h_0, m; r), y_0, \tau_{00}) = 1$ and $\mathrm{V.Verify}(pk, y_0, z_0, \tau_{01}) = 1$. And since VRF has completeness and uniqueness, we have $y_0 = \mathrm{V.Eval}(sk, (h_0, m; r))$ and $z_0 = \mathrm{V.Eval}(sk, y_0)$.

In this case, if the VRF challenger's bit $b = 0$, then we have $y = z_0$. Recall that this is the trigger condition for $\mathcal{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathcal{B}$, $\mathcal{B}$ outputs a random bit with overwhelm probability.

Furthermore, let us consider the probability that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input $x = \mathrm{VRF.Eval}(sk, (h_0, m; r))$. $\mathcal{B}$ queries its oracle only when $\mathcal{A}$ queries oracle respect input i^*. Since when $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Reun}}$, $\mathcal{A}$ cannot query its signing oracle $\mathrm{OS}(\cdot)$ on $(\cdot, m, \mathrm{R})$, and cannot query its repudiation oracle on $(i^*, m, \mathrm{R}, \cdot)$, there is only a negligible probability such that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input x.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathcal{B}) \geq \frac{1}{4l^2} \cdot \mathrm{Adv}_{\mathrm{Reun1}}(\mathcal{A}).$$

This concludes the proof of signer cannot forge repudiations for others.

Non-signer cannot forge reputations for others: Assume there exists a PPT adversary $\mathcal{A}$ that breaks our RRS scheme (in the sense of $\mathrm{Exp}_{\mathrm{Reun2}}$) with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability. The reduction is given as follows.

Adversary $\mathcal{B}$: given input 1^λ, pk, and access to oracle $\mathrm{Eval}(sk, \cdot)$, $\mathrm{Prove}(sk, \cdot)$.

1. $\mathcal{B}$ runs $\mathrm{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\mathrm{VK}_{i^*} = (pk^0_{i^*}, pk)$. Then $\mathcal{B}$ gives 1^λ and $\mathrm{VK}_1, \ldots, \mathrm{VK}_l$ to $\mathcal{A}$.
2. $\mathcal{B}$ proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:
   - If $\mathcal{A}$ queries its corruption oracle $\mathrm{OC}(\cdot)$ on a user $i \neq i^*$, $\mathcal{B}$ faithfully answers SK_i to $\mathcal{A}$. If $\mathcal{A}$ makes a corruption query for i^*, then $\mathcal{B}$ simply aborts.
   - If $\mathcal{A}$ queries its signing oracle $\mathrm{OS}(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Sign}(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathcal{A}$. If $\mathcal{A}$ makes a signing query for (i^*, m, R), then $\mathcal{B}$ runs the honest signing algorithm RRS.Sign with the following modification: in step 3, instead of using sk to generate y^1_0 and τ^1, $\mathcal{B}$ generates these by invoking its VRF oracle.
   - If $\mathcal{A}$ queries its repudiation oracle $\mathrm{OR}(\cdot)$ on $(i, m, \mathrm{R}, \sigma)$, where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Repudiate}(\mathrm{SK}_i, m, \mathrm{R}, \sigma)$ to $\mathcal{A}$. If $\mathcal{A}$ makes a repudiation query for $(i^*, m, \mathrm{R}, \sigma)$, then $\mathcal{B}$ runs the honest repudiating algorithm RRS.Repudiate with the following modification: in step 3, step 4, instead of using sk to generate y_0, y_1, τ_{00}, τ_{10}, z_0, z_1, and τ_{01}, τ_{11}, $\mathcal{B}$ generates these by invoking its VRF oracle.
3. Then $\mathcal{A}$ outputs (m, R, i), If $i = i^*$, then $\mathcal{B}$ simply aborts. Else, $\mathcal{B}$ runs the honest signing algorithm RRS.Sign and outputs $\mathrm{RRS.Sign}(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathcal{A}$;
4. $\mathcal{A}$ continues to query $\mathrm{OC}(\cdot)$ and $\mathrm{OS}(\cdot)$, and $\mathcal{B}$ answers queries in the same way described above.
5. Finally $\mathcal{A}$ outputs (j, ξ). If $j \neq i^*$, then $\mathcal{B}$ simply aborts. Else $\mathcal{B}$ parses $\sigma = (r, y^0_0, y^1_0, y^0_1, y^1_1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$ and $\xi = (z_0, z_1, \pi')$, and do:
   - Compute $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R})$;
   - $\mathcal{B}$ queries its oracle $\mathrm{Eval}(\cdot)$ on $(h_0, m; r)$ and get x.
   - $\mathcal{B}$ submits x to the VRF challenger and then receives responses y. If $y = z_0$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

It remains to show that the adversary $\mathcal{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. First note that the distribution (i.e., verification keys and oracle responses) of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathcal{A}$ does not trigger this abort. And, conditioned that $\mathcal{A}$ does not ask to corrupt i^*, from the view of $\mathcal{A}$ the index i^* is distributed uniformly random, so it holds that $i \neq i^*$ with probability at least $\frac{1}{l}$. Also by the randomness of i^*, we also have that $j = i^*$ with probability at least $\frac{1}{l}$.

Now, we assume that no abort happened and $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Reun2}}$. Then ξ will be a valid repudiation of i^* respect to σ, i.e.

$$\mathrm{RRS.VerRepud}(\mathrm{VK}_{i^*}, m, \mathrm{R}, \sigma, \xi) = 1.$$

Therefore, by the soundness of NIWI, there must exist y_0, and τ_{00}, τ_{01}, s.t. we have $\mathrm{V.Verify}(pk, (h_0, m; r), y_0, \tau_{00}) = 1$ and $\mathrm{V.Verify}(pk, y_0, z_0, \tau_{01}) = 1$. And since VRF has completeness and uniqueness, we have $y_0 = \mathrm{V.Eval}(sk, (h_0, m; r))$ and $z_0 = \mathrm{V.Eval}(sk, y_0)$.

In this case, if the VRF challenger's bit $b = 0$, then we have $y = z_0$. Recall that this is the trigger condition for $\mathcal{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathcal{B}$, $\mathcal{B}$ outputs a random bit with overwhelm probability.

Furthermore, let us consider when $\mathcal{E}_2$ occurs, the probability that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input $x = \mathrm{VRF.Eval}(sk, (h_0, m, \varphi))$. $\mathcal{B}$ queries its oracle only when $\mathcal{A}$ queries oracle respect input i^*. When $\mathcal{A}$ queries its signature oracle, $\mathcal{B}$ will chooses a random strings φ, so there is only negligible probability such that $\mathcal{B}$ meets x. And since σ is generated by $\mathcal{B}$, and after signature produced $\mathcal{A}$ cannot query its repudiation oracle, there is only negligible probability such that $\mathcal{A}$ queries a repudiation oracle with a signature which include φ. Therefore, there is only a negligible probability such that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input x.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathcal{B}) \geq \frac{1}{4l^3} \cdot \mathrm{Adv}_{\mathrm{Reun2}}(\mathcal{A}).$$

This concludes the proof.

4.5 Anonymity

We will now turn to establish the anonymity of our RRS scheme.

Theorem 3. *The repudiable ring signature scheme* RRS *satisfies anonymity against adversarially chosen keys, given that* NIWI *has soundness and witness indistinguishablity,* VRF *has commpletenss, uniqueness, and pseudorandomness, and* SPB *has index hiding.*

The main idea of the proof is that, first move the index of hk_1 from j_0 to j_1 and argue indistinguishability via the index-hiding property of SPB. Next we switch y_1^0, y_1^1 to the evaluation of $(h_1, m; r)$ and $(h_1, m; r)$, where h_1 is the digest of R for new key hk_1. This modification will not be detected due to the pseudorandom property of VRF. Now, we can switch the NIWI witness to $(\mathrm{VK}_{j_1}, \mathrm{ind}_1, \eta_1, \tau_1^0, \tau_1^1, 1)$, and by witness indistinguishability of NIWI, this signature also satisfies indistinguishability. Next, we perform the first two changes above for hk_0 and y_0^0, y_0^1, switch the witness back to the witness for $j = 0$, and finally replace y_1^0, y_1^1 with a random string. The signature in the last experiment is now a real signature of m under VK_{j_1}.

Proof. Let $\mathcal{A}$ be a PPT adversary against the anonymity of RRS. Assume that $\mathcal{A}$ makes at most $q = \text{poly}(\lambda)$ queries for any oracle. Let in the following ind_0 be the index of VK_{j_0} in R, and ind_1 be the index of VK_{j_1} in R, where (m, R, j_0, j_1) is the challenge query of $\mathcal{A}$. Now, consider the following hybrids:

Hybrid 1: This is the real experiment with challenge bit $b = 0$.
Hybrid 2: Same as Hybrid 1, except that in σ, we compute hk_1 by $(\text{hk}_1, \text{shk}_1) \leftarrow \text{S.Gen}(1^\lambda, |\text{R}|, \text{ind}_1)$.

Hybrid 1 and Hybrid 2 are computationally indistinguishable, given that SPB is index hiding. More specifically, there exists a reduction $\mathcal{R}_1$ such that

$$\text{Adv}_{\text{Index-Hiding}}(\mathcal{R}_1^{\mathcal{A}}) = \text{Adv}_{\text{H}_1,\text{H}_2}(\mathcal{A}).$$

We will provide an informal description of $\mathcal{R}_1$. The $\mathcal{R}_1$ simulates H_1 faithfully, until $\mathcal{A}$ outputs a challenger query (m, R, j_0, j_1). Then $\mathcal{R}_1$ gives $(|\text{R}|, \text{ind}_0, \text{ind}_1)$ to the index hiding experiment and receives a hashing key hk^*. $\mathcal{R}_1$ continues the simulation of H_1 faithfully, except that in the challenge signature it sets $\text{hk}_1 = \text{hk}^*$. In the end, $\mathcal{R}_1$ outputs the output of $\mathcal{A}$.

Clearly, if the challenge bit of the index hiding experiment is 0 then $\mathcal{R}_1$ simulates Hybrid 1 perfectly. And if the challenge bit of the index hiding experiment is 1 then $\mathcal{R}_1$ simulates Hybrid 2 perfectly. Therefore, Hybrid 1 and Hybrid 2 are computationally indistinguishable.

Hybrid 3: Same as Hybrid 2, except that we compute y_1^1 by $y_1^1 = \text{V.Eval}(sk_{j_1}^1, (h_1, m; r))$.

Hybrid 2 and Hybrid 3 are computationally indistinguishable, given that the VRF is pseudorandom. More specifically, there exists a reduction $\mathcal{R}_2$ such that

$$\text{Adv}_{\text{VRF}}(\mathcal{R}_2^{\mathcal{A}}) = \text{Adv}_{\text{H}_2,\text{H}_3}(\mathcal{A}) \cdot \text{poly}(\lambda).$$

The proof is very similar with the proof of Theorem 2. We only provide an informal description of $\mathcal{R}_2$. The reduction $\mathcal{R}_2$ receives as input a public key pk. The $\mathcal{R}_2$ simulates H_2 faithfully, except for the following. Before the simulation stars, $\mathcal{R}_2$ chooses a random index i^* and sets $\text{VK}_{i^*} = (pk_{i^*}^0, pk)$, where $pk_{i^*}^0$ is generated as in H_2 and pk is the input of $\mathcal{R}_2$. $\mathcal{R}_2$ continues the simulation of H_2 [6] until $\mathcal{A}$ announces (j_0, j_1, m, R). If it holds $j_1 \neq i^*$, $\mathcal{R}_2$ outputs $\perp$. Otherwise, $\mathcal{R}_2$ continues the simulation of H_2 faithfully, except that in the challenge signature it sets $y_1^1 = y$, where y is the outputs of challenger when $\mathcal{B}$ gives it (h_1, m, φ). Finally, $\mathcal{R}_2$ continues the simulation and outputs whatever $\mathcal{A}$ outputs.

Clearly, if the challenge bit of the VRF experiment is 0 then $\mathcal{R}_1$ simulates Hybrid 2 perfectly. And if the challenge bit of the index hiding experiment is 1 then $\mathcal{R}_1$ simulates Hybrid 3 perfectly. And since we require $\mathcal{A}$ can not query $\text{OR}(\cdot)$ with $(\cdot, m, \text{R}, \cdot)$, after he has received a challenge signature, $\mathcal{A}$ can not get any advantage from his repudiation oracle. Therefore, Hybrid 2 and Hybrid 3 are computationally indistinguishable.

[6] When $\mathcal{A}$ queries its oracle, the answer of $\mathcal{R}_2$ is same as $\mathcal{B}$'s answer in Theorem 2.

Hybrid 4: Same as Hybrid 3, except that we compute y_1^0 by $y_1^0 = \text{V.Eval}(sk_{j_1}^0, (h_1, m; r))$.

Hybrid 3 and Hybrid 4 are computationally indistinguishable, given that the VRF is pseudorandom. More specifically, there exists a reduction $\mathcal{R}_3$ such that

$$\text{Adv}_{\text{VRF}}(\mathcal{R}_3^{\mathcal{A}}) = \text{Adv}_{\text{H}_3,\text{H}_4}(\mathcal{A}) \cdot \text{poly}(\lambda).$$

The proof is very similar to the above, except that we set $\text{VK}_{i^*} = (pk, pk_{i^*}^1)$, where $pk_{i^*}^1$ is generated as in H_3 and pk is the input of $\mathcal{R}_3$.

Hybrid 5: Same as Hybrid 4, except that we compute witness by
- $\eta \leftarrow \text{S.Open}(\text{hk}_1, \text{shk}_1, \text{R}, \text{ind}_1)$,
- $\tau^0 \leftarrow \text{V.Prove}(sk_{j_1}^0, (h_1, m; r))$, $\tau^1 \leftarrow \text{VRF.Prove}(sk_{j_1}^1, (h_1, m; r))$,

and use the witness $w = (\text{VK}_{j_1}, \text{ind}_1, \eta, \tau^0, \tau^1, 1)$ to compute π.

Hybrid 4 and Hybrid 5 are computationally indistinguishable, give that NIWI is computationally witness indistinguishable. More specifically, there exists a reduction $\mathcal{R}_4$ against the witness indistinguishability of NIWI such that

$$\text{Adv}_{\text{WI}}(\mathcal{R}_4^{\mathcal{A}}) = \text{Adv}_{\text{H}_4,\text{H}_5}(\mathcal{A}).$$

The reduction $\mathcal{R}_4$ simulates H_4 faithfully, until the challenge signature is computed. Instead of computing the proof π itself, $\mathcal{R}_4$ sends the statement $x = (m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1)$ and the witness $w_0 = (\text{VK}_{j_0}, \text{ind}_0, \eta_0, \tau_0^0, \tau_0^1, 0)$ and $w_1 = (\text{VK}_{j_1}, \text{ind}_1, \eta_1, \tau_1^0, \tau_1^1, 1)$ to the witness indistinguishability experiment. The experiment returns a proof π^*, and $\mathcal{R}_4$ use the proof π^* in the challenge signature. $\mathcal{R}_4$ continues the simulation of H_4 faithfully and outputs whatever the simulated $\mathcal{A}$ outputs.

Clearly, if the challenge bit of the witness indistinguishability experiment is 0, then $\mathcal{R}_4$ simulates H_4 perfectly. On the other hand, if the challenge bit is 1, then $\mathcal{R}_4$ simulates H_5 perfectly. Therefore, Hybrid 4 and Hybrid 5 are computationally indistinguishable.

Next, we perform the similar changes as following.

Hybrid 6: Same as Hybrid 5, except that we compute y_0^0 by $y_0^0 \leftarrow \{0,1\}^{\alpha(\lambda)}$.
Hybrid 7: Same as Hybrid 6, except that we compute y_0^1 by $y_0^1 \leftarrow \{0,1\}^{\alpha(\lambda)}$.
Hybrid 8: Same as Hybrid 7, except that in σ, we compute hk_0 by $(\text{hk}_0, \text{shk}_0) \leftarrow \text{SPB.Gen}(1^\lambda, |\text{R}|, \text{ind}_1)$.
Hybrid 9: Same as Hybrid 8, except that we compute y_0^0 by $y_0^0 = \text{V.Eval}(sk_{j_1}^0, (h_0, m; r))$.
Hybrid 10: Same as Hybrid 9, except that we compute y_0^1 by $y_0^1 = \text{V.Eval}(sk_{j_1}^1, (h_0, m; r))$.
Hybrid 11: Same as Hybrid 10, except that we compute witness by
- $\eta \leftarrow \text{SPB.Open}(\text{hk}_0, \text{shk}_0, \text{R}, \text{ind}_1)$,
- $\tau^0 \leftarrow \text{VRF.Prove}(sk_{j_1}^0, (h_0, m, \varphi))$, $\tau^1 \leftarrow \text{VRF.Prove}(sk_{j_1}^1, (h_0, m, \varphi))$,

and use the witness $w = (\text{VK}_{j_1}, \text{ind}_1, \eta, \tau^0, \tau^1, 0)$ to compute π.
Hybrid 12: The same as hybrid 11, except that we compute y_1^0 and y_1^1 by $y_1^0, y_1^1 \leftarrow \{0,1\}^{\alpha(\lambda)}$. This is identical to the real experiment with $b = 1$.

These hybrids are also indistinguishability, the proof is analogous, and we omit it. Therefore, RRS satisfies anonymity against adversarially chosen keys.

4.6 Unforgeability

We will now turn to show that our RRS scheme is unforgeable.

Theorem 4. *The repudiable ring signature scheme* RRS *is unforgeable, given that* NIWI *has soundness,* VRF *has completeness, uniqueness, and pseudorandomness, and* SPB *has somewhere perfectly binding.*

The main idea of the proof is that, if $\mathcal{A}$ can forge a valid signature, then since NIWI has soundness, it must has VK, $i \in [N]$, η, τ^0, τ^1, and $j \in \{0,1\}$ s.t. we have $\text{S.Verify}(\text{hk}_j, h_j, i, \text{VK}, \eta) = 1$, $\text{V.Verify}(pk^0, (h_j, m; r), y_j^0, \tau^0) = 1$, and $\text{V.Verify}(pk^1, (h_j, m; r), y_j^1, \tau^1) = 1$. Since SPB has somewhere perfectly binding, we have $\text{VK} = \text{R}[i]$. And by the completeness and uniqueness of the VRF, we have $y_j^0 = \text{V.Eval}(pk^0, (h_j, m, r))$, and $y_j^1 = \text{V.Eval}(pk^1, (h_j, m, r))$, and by this we can attack the pseudorandomness of VRF.

Proof. Assume there exists a PPT adversary $\mathcal{A}$ that breaks our RRS scheme (in the sense of Definition 7) with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability. The reduction is given as follows.

Adversary $\mathcal{B}$: given input 1^λ, pk, and access to oracle $\text{Eval}(sk, \cdot)$, $\text{Prove}(sk, \cdot)$.

1. $\mathcal{B}$ runs $\text{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\text{VK}_{i^*} = (pk, pk_{i^*}^1)$. Then $\mathcal{B}$ gives 1^λ and $\text{VK}_1, \ldots, \text{VK}_l$ to $\mathcal{A}$.
2. $\mathcal{B}$ proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:
   - If $\mathcal{A}$ queries its corruption oracle $\text{OC}(\cdot)$ on a user $i \neq i^*$, $\mathcal{B}$ faithfully answers SK_i to $\mathcal{A}$. If $\mathcal{A}$ makes a corruption query for i^*, then $\mathcal{B}$ simply aborts.
   - If $\mathcal{A}$ queries its signing oracle $\text{OS}(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\text{RRS.Sign}(\text{SK}_i, m, \text{R})$ to $\mathcal{A}$. If $\mathcal{A}$ makes a signing query for (i^*, m, R), then $\mathcal{B}$ runs the honest signing algorithm RRS.Sign with the following modification: in step 3, instead of using sk to generate y_0^0 and τ^0, $\mathcal{B}$ generates these by invoking its VRF oracle.
   - If $\mathcal{A}$ queries its repudiation oracle $\text{OR}(\cdot)$ on (j, m, R, σ), $\mathcal{B}$ faithfully answers $\text{RRS.Repudiate}(\text{SK}_i, m, \text{R}, \sigma)$ to $\mathcal{A}$. [7]
3. Finally $\mathcal{A}$ outputs (m, R, σ). Then $\mathcal{B}$ computes $t = \text{RRS.Verify}(m, \text{R}, \sigma)$. If $t = 0$, $\mathcal{B}$ simply aborts. Else $\mathcal{B}$ parses $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, \pi)$, and do :
   - Compute $h_0 = \text{S.Hash}(\text{hk}_0, \text{R})$, $h_1 = \text{S.Hash}(\text{hk}_1, \text{R})$;
   - Choose a random bit $k \leftarrow \{0,1\}$.
   - $\mathcal{B}$ submits $(h_k, m; r)$ to the VRF challenger and then receive responses y. If $y = y_k^0$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

[7] Since sk is not used by RRS.Repudiate, $\mathcal{B}$ does not need to invoke the VRF oracle here.

It remains to show that the adversary $\mathcal{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. First note that the distribution (i.e., verification keys and oracle responses) of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathcal{A}$ does not trigger this abort.

Now, we assume that $\mathcal{A}$ does not query corruption oracle on i^* and $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Reun}}$. Then σ is a valid signature of m with respect to R, i.e.

$$\mathrm{RRS.Verify}(m, \mathrm{R}, \sigma) = 1.$$

Therefore, by the soundness of NIWI, there exists VK, i, η, τ^0, τ^1, and $j \in \{0,1\}$ s.t. it holds $\mathrm{S.Verify}(\mathrm{hk}_j, h_j, i, \mathrm{VK}, \eta) = 1$, $\mathrm{V.Verify}(pk^0, (h_j, m; r), y_j^0, \tau^0) = 1$ and $\mathrm{V.Verify}(pk^1, (h_j, m; r), y_j^1, \tau^1) = 1$. Since SPB has somewhere perfectly binding, we have $\mathrm{VK} = \mathrm{R}[i]$. And by the completeness and uniqueness of the VRF, we have $y_j^0 = \mathrm{V.Eval}(sk^0, (h_j, m; r))$, and $y_j^1 = \mathrm{V.Eval}(sk^1, (h_j, m; r))$.

And when $i = i^*$, and $j = k$, happened, we have $y_k^0 = \mathrm{V.Eval}(sk, (h_k, m; r))$. In this case, if the VRF challenger's bit $b = 0$, then we have $y = y_k^0$. Recall that this is the trigger condition for $\mathcal{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is turly random strings. Thus, by the definition of $\mathcal{B}$, $\mathcal{B}$ outputs a random bit with overwhelm probability.

Now let us consider the probability that $i = i^*$ and $j = k$ occurs condition on no abort happened. As also observed above, the distribution of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$, $i = i^*$ occurs. And since k is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{2}$, $j = k$ occurs.

Furthermore, let us consider the probability that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input $(h_k, m; r)$. $\mathcal{B}$ queries its oracle only when $\mathcal{A}$ queries his signing oracle respect input i^*. Since when $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Reun}}$, $\mathcal{A}$ cannot query its signing oracle $\mathrm{OS}(\cdot)$ on $(\cdot, m, \mathrm{R})$, there is only a negligible probability such that $\mathcal{B}$ queries oracle $\mathrm{Eval}(sk, \cdot)$ or $\mathrm{Prove}(sk, \cdot)$ for input $(h_k, m; r)$.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathcal{B}) \geq \frac{1}{8l^2} \cdot \mathrm{Adv}_{\mathrm{Unf}}(\mathcal{A}).$$

This concludes the proof.

4.7 Repudiability

We will now turn to show that our scheme is repudiable.

Theorem 5. *The repudiable ring signature scheme* RRS *is repudiable, given that* NIWI *has completeness, soundness,* VRF *has completeness, uniqueness, pseudorandomness, and* SPB *has somewhere perfectly binding and index hiding.*

The main idea of the proof is that, by the definition of repudiability, we need to proof non-signer can repudiate and signer cannot repudiate separately. The proof of non-signer can repudiate is that, if there is an honeset non-signer cannot repudiate, then by the soundness of NIWI, we have $y_0 = y_0^1$ or $y_1 = y_1^1$ hold, and by this we can attack the pseudorandomness of VRF. The proof of signer cannot repudiate is that, if $\mathcal{A}$ can produce a valid repudiation, then by the soundness of NIWI, we have that $\mathcal{A}$ forges a valid signature for others, and so we can attack the pseudorandomess of VRF.

Proof. We will prove each of the desired security properties in turn.

Non-signer can repudiate: Assume there exists a PPT adversary $\mathcal{A}$ that breaks our RRS scheme (in the sense of $\mathrm{Exp}_{\mathrm{Rep1}}$) with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability. The reduction is given as follows.

Adversary $\mathcal{B}$: given input 1^λ, pk, and access to oracle $\mathrm{Eval}(sk,\cdot)$, $\mathrm{Prove}(sk,\cdot)$.

1. $\mathcal{B}$ runs $\mathrm{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\mathrm{VK}_{i^*} = (pk^0_{i^*}, pk)$. Then $\mathcal{B}$ gives 1^λ and $\mathrm{VK}_1, \ldots, \mathrm{VK}_l$ to $\mathcal{A}$.
2. $\mathcal{B}$ proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:
   - If $\mathcal{A}$ queries its corruption oracle $\mathrm{OC}(\cdot)$ on a user $i \neq i^*$, $\mathcal{B}$ faithfully answers SK_i to $\mathcal{A}$. If $\mathcal{A}$ makes a corruption query for i^*, then $\mathcal{B}$ simply aborts.
   - If $\mathcal{A}$ queries its signing oracle $\mathrm{OS}(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Sign}(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathcal{A}$. If $\mathcal{A}$ makes a signing query for (i^*, m, R), then $\mathcal{B}$ runs the honest signing algorithm RRS.Sign with the following modification: in step 3, instead of using sk to generate y_0^1 and τ^1, $\mathcal{B}$ generates these by invoking its VRF oracle.
   - If $\mathcal{A}$ queries its repudiation oracle $\mathrm{OR}(\cdot)$ on $(i, m, \mathrm{R}, \sigma)$, where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Repudiate}(\mathrm{SK}_i, m, \mathrm{R}, \sigma)$ to $\mathcal{A}$. If $\mathcal{A}$ makes a repudiation query for $(i^*, m, \mathrm{R}, \sigma)$, then $\mathcal{B}$ runs the honest repudiating algorithm RRS.Repudiate with the following modification: in step 3, step 4, instead of using sk to generate y_0, y_1, τ_{00}, τ_{10}, z_0, z_1, and τ_{01}, τ_{11}, $\mathcal{B}$ generates these by invoking its VRF oracle.
3. Finally $\mathcal{A}$ outputs (m, R, σ). Then $\mathcal{B}$ parse $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, and do:
   - Compute $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R})$, $h_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R})$;
   - Choose a random bit $k \leftarrow \{0, 1\}$.
   - $\mathcal{B}$ submits $(h_k, m; r)$ to the VRF challenger and then receive responses y. If $y = y_k^1$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

It remains to show that the adversary $\mathcal{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. First note that the distribution (i.e., verification keys and oracle responses) of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathcal{A}$ does not trigger this abort.

Now, we assume that $\mathcal{A}$ does not query corruption oracle on i^* and $\mathcal{A}$ wins the experiment Exp_{Rep1}. Then by the definition of experiment Exp_{Rep1}, we have RRS.VerRepud will reject on an honestly generated repudiation ξ_i, generated with respect to VK_i, i.e. $\exists i \in \text{R} \setminus \mathcal{Q}_{\text{OC}}$, s.t.

$$\text{VerRepud}(\text{VK}_i, m, \text{R}, \sigma, \xi_i) = 0, \text{ where } \xi_i \leftarrow \text{Repudiate}(\text{SK}_i, m, \text{R}, \sigma).$$

Since ξ_i is honestly generated with respect to VK_i, and by the definition of RRS.Repudiate and RRS.VerRepud, we have there exist $j \in \{0, 1\}$ s.t. $y_j = y_j^1$, where $y_j = \text{V.Eval}(sk_i^1, (\tilde{h}_j, m; r))$. Since if the above formula are not true, then by the completeness of the NIWI and the completeness of the VRF, we have $\text{VerRepud}(\text{VK}_i, m, \text{R}, \sigma, \xi_i) = 1$.

And when $i = i^*$, and $j = k$ happens, we have $y_k^1 = \text{V.Eval}(sk, (h_k, m; r))$. In this case, if the VRF challenger's bit $b = 0$, then we have $y = y_k^1$. Recall that this is the trigger condition for $\mathcal{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is turly random strings. Thus, by the definition of $\mathcal{B}$, $\mathcal{B}$ outputs a random bit with overwhelm probability.

Now let us consider the probability that $i = i^*$ and $j = k$ occurs condition on no abort happened. As also observed above, the distribution of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$, $i = i^*$ occurs. And since k is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{2}$, $j = k$ occurs.

Furthermore, let us consider the probability that $\mathcal{B}$ queries oracle $\text{Eval}(sk, \cdot)$ or $\text{Prove}(sk, \cdot)$ for input (h_k, m, r). $\mathcal{B}$ queries its oracle only when $\mathcal{A}$ queries oracle respect input i^*. Since when $\mathcal{A}$ wins the experiment Exp_{Reun}, $\mathcal{A}$ cannot query its signing oracle $\text{OS}(\cdot)$ on $(\cdot, m, \text{R})$, and cannot query its repudiation oracle on $(\cdot, m, \text{R}, \cdot)$, there is only a negligible probability such that $\mathcal{B}$ queries oracle $\text{Eval}(sk, \cdot)$ or $\text{Prove}(sk, \cdot)$ for input (h_k, m, r).

All together, we can conclude that

$$\text{Adv}_{\text{VRF}}(\mathcal{B}) \geq \frac{1}{8l^2} \cdot \text{Adv}_{\text{Rep1}}(\mathcal{A}).$$

This concludes the proof of non-signer can repudiate.

Signer cannot repudiate: Assume there exists a PPT adversary $\mathcal{A}$ that breaks our RRS scheme (in the sense of Exp_{Rep2}) with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability. The reduction is given as follows.

Adversary $\mathcal{B}$: given input 1^λ, pk, and access to oracle $\text{Eval}(sk, \cdot)$, $\text{Prove}(sk, \cdot)$.

1. $\mathcal{B}$ runs $\text{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\text{VK}_{i^*} = (pk, pk_{i^*}^1)$. Then $\mathcal{B}$ gives 1^λ and $\text{VK}_1, \ldots, \text{VK}_l$ to $\mathcal{A}$.
2. $\mathcal{B}$ proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:

  - If $\mathcal{A}$ queries its corruption oracle $\mathrm{OC}(\cdot)$ on a user $i \neq i^*$, $\mathcal{B}$ faithfully answers SK_i to $\mathcal{A}$. If $\mathcal{A}$ makes a corruption query for i^*, then $\mathcal{B}$ simply aborts.
  - If $\mathcal{A}$ queries its signing oracle $\mathrm{OS}(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Sign}(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathcal{A}$. If $\mathcal{A}$ makes a signing query for (i^*, m, R), then $\mathcal{B}$ runs the honest signing algorithm RRS.Sign with the following modification: in step 3, instead of using sk to generate y_0^0 and τ^0, $\mathcal{B}$ generates these by invoking its VRF oracle.
  - If $\mathcal{A}$ queries its repudiation oracle $\mathrm{OR}(\cdot)$ on $(j, m, \mathrm{R}, \sigma)$, $\mathcal{B}$ faithfully answers $\mathrm{RRS.Repudiate}(\mathrm{SK}_i, m, \mathrm{R}, \sigma)$ to $\mathcal{A}$.
3. Finally $\mathcal{A}$ outputs $(m, \mathrm{R}, \sigma, \{\xi_{i_k}\}_{\mathrm{VK}_{i_k} \in \mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R}})$. $\mathcal{B}$ runs the honest verifying algorithm RRS.Verify on (m,R, σ). If $\mathrm{RRS.Verify}(m,\ \mathrm{R}, \sigma) = 0$, $\mathcal{B}$ simply aborts. Then $\mathcal{B}$ runs the honest verrepud algorithm RRS.VerRepud, $b_{i_k} = \mathrm{RRS.VerRepud}(\mathrm{VK}_{i_k}, m, \mathrm{R}, \sigma, \xi_{i_k})$ for all $\mathrm{VK}_{i_k} \in \mathcal{Q}_{\mathrm{OC}} \cap \mathrm{R}$. If there exists a $b_{i_k} = 0$, $\mathcal{B}$ also simply aborts. Else $\mathcal{B}$ parse $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, and do :
   - Compute $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R})$, $h_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R})$;
   - Choose a random bit $k \leftarrow \{0, 1\}$.
   - $\mathcal{B}$ submits $(h_k, m; r)$ to the VRF challenger and then receive responses y. If $y = y_k^0$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

It remains to show that the adversary $\mathcal{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. First we note that, if $\mathcal{A}$ queries all keys, then we have $\mathrm{R} \subset \mathcal{Q}_{OC}$, in this situation we can proof $\mathcal{A}$ cannot win the game $\mathrm{Exp}_{\mathrm{Rep2}}$. Since in this case, if $\mathcal{A}$ wins the game, then σ will be a valid signature for m with respect to R. By soundness of NIWI, there exist VK, $i \in [N]$, η, τ^0, τ^1, and $j \in \{0, 1\}$ s.t. it holds $\mathrm{S.Verify}(\mathrm{hk}_j, h_j, i, \mathrm{VK}, \eta) = 1$ and $\mathrm{V.Verify}(pk^0, (h_j, m; r), y_j^0, \tau^0) = 1$, and $\mathrm{V.Verify}(pk^1, (h_j, m; r), y_j^1, \tau^1) = 1$. By somewhere perfectly binding of SPB, we have $\mathrm{VK} = \mathrm{R}[i]$. And by the completeness and uniqueness of the VRF, we have $y_j^0 = \mathrm{V.Eval}(sk^0, (h_j, m; r))$, and $y_j^1 = \mathrm{V.Eval}(sk^1, (h_j, m; r))$. Now, let us consider the repudiation of VK, if ξ is a valid repudiation. Then by soundness of NIWI, there exist y_0, y_1 τ_{00}, τ_{01}, τ_{10}, τ_{11}, s.t. it holds $\mathrm{V.Verify}(pk^1, (h_0, m; r), y_0, \tau_{00}) = 1$, and $\mathrm{V.Verify}(pk^1, y_0', z_0, \tau_{01}) = 1$, $\mathrm{V.Verify}(pk^1, (h_1, m; r), y_1, \tau_{10}) = 1$, and $\mathrm{VRF.Verify}(pk^1, y_1, z_1, \tau_{11}) = 1$. Since VRF has completeness and uniqueness, we have $y_0 = \mathrm{V.Eval}(pk^1, (h_0, m; t))$, and $z_0 = \mathrm{V.Eval}(pk^1, y_0)$, $y_1 = \mathrm{V.Eval}(pk^1, (h_1, m; r))$, and $z_1 = \mathrm{V.Eval}(pk^1, y_1)$. Therefore, by the uniqueness of VRF, we have $(y_0 = y_0^1) \vee (y_1 = y_1^1) = 1$ with overwhelm probability. And this is contradictory.

So when $\mathcal{A}$ wins the game, he must leave one or more keys uncorrupted in R. Since the distribution (i.e., verification keys and oracle responses) of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. And Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathcal{A}$ does not trigger this abort.

Now, we assume that $\mathcal{A}$ does not query corruption oracle on i^* and $\mathcal{A}$ wins the experiment $\mathrm{Exp}_{\mathrm{Rep2}}$. Then by the definition of experiment $\mathrm{Exp}_{\mathrm{Rep2}}$, we have σ is a valid signature of m with respect to R and ξ_{i_k} are all valid repudiation. Then by soundness of NIWI, somewhere perfectly binding of SPB and completeness and uniqueness of VRF, we have there exist VK, i, and $j \in \{0,1\}$ such that $\mathrm{VK} = \mathrm{R}[i]$, $y_j^0 = \mathrm{V.Eval}(pk^0, (h_j, m; r))$, and $y_j^1 = \mathrm{V.Eval}(pk^1, (h_j, m; r))$. And according to the above discussion, we have $\mathrm{VK} \notin \mathcal{Q}_{OC}$.

And when $i = i^*$, and $j = k$ happens, we have $y_k^0 = \mathrm{V.Eval}(sk, (h_k, m; r))$. In this case, if the VRF challenger's bit $b = 0$, then we have $y = y_k^0$. Recall that this is the trigger condition for $\mathcal{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is turly random strings. Thus, by the definition of $\mathcal{B}$, $\mathcal{B}$ outputs a random bit with overwhelm probability. [8]

Now let us consider the probability that $i = i^*$ and $j = k$ occurs condition on no abort happened. As also observed above, the distribution of the view of $\mathcal{A}$ is unaffected by $\mathcal{B}$'s choice of i^*, until the point at which $\mathcal{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{l}$, $i = i^*$ occurs. And since k is chosen at random by $\mathcal{B}$, it follows that with probability at least $\frac{1}{2}$, $j = k$ occurs.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathcal{B}) \geq \frac{1}{8l^2} \cdot \mathrm{Adv}_{\mathrm{Rep2}}(\mathcal{A}).$$

This concludes the proof of signer cannot repudiate.

5 Conclusions

Ring signature is a well-studied cryptographic primitive with many applications. Repudiable ring signature is a more stronger cryptographic primitive than ring signature, which can allow non-signer repudiate a signature that was not produced by him. In this paper we improved the security definition of repudiable ring signature, and proposed a new requirement repudiation unforgeability. Beside, we also proposed a new scheme satisfied our definiton. This scheme has that signature and repudiation size is logarithmic in the number of ring members, while at the same time relying on standard assumptions and not requiring a trusted setup.

References

1. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setupfrom standard assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 281–311. Springer (2019)
2. Barak, B., Ong, S.J., Vadhan, S.: Derandomization in cryptography. In: Annual International Cryptology Conference. pp. 299–315. Springer (2003)

[8] Besides, in this case, the probability that $\mathcal{B}$ queries its oracle on (h_k, m, φ) is negligible.

3. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Theory of Cryptography Conference. pp. 401–427. Springer (2015)
4. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on ddh. In: European Symposium on Research in Computer Security. pp. 243–265. Springer (2015)
5. Dodis, Y.: Efficient construction of (distributed) verifiable random functions. In: International Workshop on Public Key Cryptography. pp. 1–17. Springer (2003)
6. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 609–626. Springer (2004)
7. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: International Workshop on Public Key Cryptography. pp. 416–431. Springer (2005)
8. Dwork, C., Naor, M.: Zaps and their applications. In: Proceedings 41st Annual Symposium on Foundations of Computer Science. pp. 283–293. IEEE (2000)
9. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
10. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. Journal of the ACM (JACM) **59**(3), 11 (2012)
11. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science. pp. 163–172. ACM (2015)
12. Libert, B., Peters, T., Qian, C.: Logarithmic-size ring signatures with tight security from the ddh assumption. In: European Symposium on Research in Computer Security. pp. 288–308. Springer (2018)
13. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
14. Lysyanskaya, A.: Unique signatures and verifiable random functions from the dh-ddh separation. In: Annual International Cryptology Conference. pp. 597–612. Springer (2002)
15. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039). pp. 120–130. IEEE (1999)
16. Noether, S.: Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive **2015**, 1098 (2015)
17. Okamoto, T., Pietrzak, K., Waters, B., Wichs, D.: New realizations of somewhere statistically binding hashing and positional accumulators. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 121–145. Springer (2015)
18. Park, S., Sealfon, A.: It wasnt me! In: Annual International Cryptology Conference. pp. 159–190. Springer (2019)
19. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–565. Springer (2001)
20. Xu, S., Yung, M.: Accountable ring signatures: a smart card approach. In: Smart Card Research and Advanced Applications VI, pp. 271–286. Springer (2004)