Computer Standards & Interfaces 80 (2022) 103562

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Repudiable ring signature: Stronger security and logarithmic-size

Hao Lin [a,b], Mingqiang Wang [*,a,b]

[a] School of Mathematics, Shandong University, Jinan, Shandong 250100, China
[b] China Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Jinan, Shandong 250100, China

ARTICLE INFO

Keywords:
Ring Signature
Repudiable Ring Signature
Blockchain

ABSTRACT

Ring signature, introduced by Rivest et al. [Asiacrypt'01], allows a person to sign a document on behalf of an ad-hoc group (or ring) while hiding the identity of the actual signer. But the anonymity provided by the ring signature scheme can be used to conceal a malicious signer and put other ring members under suspicion. Fortunately, Park et al. [Crypto'19] proposed a repudiable ring signature scheme which can overcome this disadvantage. However, the construction of Park et al. [Crypto'19] is not compact, in other word, the size of signatures and repudiations in their scheme increases with the **square** of the ring size.

In this paper, we propose the first **logarithmic-size** repudiable ring signature scheme, which means the size of signatures and repudiations grows only logarithmically in the ring size. Moreover, in terms of security model, we present a new requirement (**repudiation-unforgeability**), which requires 'no one can forge a valid repudiation'. Our scheme is the first repudiable ring signature scheme satisfies this new requirement.

1. Introduction

Ring signature, introduced by Rivest et al. [1], 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 the signer is. 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 given the ring R. The feature of interest of the ring signature scheme is that given such a signature, no one can tell which signing key was used to compute this signature.

The original motivation for the ring signature scheme was anonymous leakage of secrets. Suppose a high rank officer wants to leak some sensitive document to a journalist without revealing its identity. To to so, he signs this document using a ring signature scheme where the ring R contains all other high rank officers. Then the journalist is convinced that some high rank officer signed this document, but he has no clue who the signer is.

More recently, the ring signature scheme has also found some applications in the construction of confidential transactions for cryptocurrencies. In a usual (non-anonymous) transaction, a user computes a signature that assesses if he is allowed to spend coins. In cryptocurrencies like Monero [2], a user forms a ring R from verification keys in the blockchain to issue a ring signature on the transaction. Thereby, the anonymity property of the ring signature scheme guarantees the confidentiality of the transaction. Currently, Monero uses a setup-free Schnorr based ring signature scheme [3].

However, providing a complete anonymity in the ring signature scheme may not always be desirable. Ring signature scheme may be open to abuse, where a malicious signer can use the anonymity to supply false information and puts other ring members under suspicion [4–7].

Deniable ring signature, introduced by Komano et al. [4], can slightly overcome this disadvantage. But the deniable ring signature scheme is an interactive protocol. Interactivity means that the *confirmation* and *disavowal* algorithms in the deniable ring signature scheme are two interactive protocols executed between a ring member and a verifier. But the interactive protocol is undesirable in most applications.[1]

Fortunately, Park et al. [7] proposed a *repudiable ring signature* scheme which can overcome these disadvantages. Repudiable ring signature is an extension of the concept of ring signature which allows a non-signer to prove to others that some signature was not generated by him. More specifically, the repudiable ring signature scheme is a ring signature scheme equipped with an additional pair of algorithms ($\mathsf{Repudiate}$, $\mathsf{VerRepud}$), where $\mathsf{Repudiate}$ is an algorithm which can create

* Corresponding author.
E-mail addresses: lhao17@mail.sdu.edu.cn (H. Lin), wangmingqiang@sdu.edu.cn (M. Wang).

[1] Although Gao et al. made the first attempt on the non-interactive deniable ring signature scheme in [6], Jia et al. recently demonstrated in [8] that the ring signature scheme in [6] is insecure.

https://doi.org/10.1016/j.csi.2021.103562
Received 6 April 2021; Received in revised form 6 July 2021; Accepted 31 July 2021
Available online 5 August 2021

Table 1
This table compares previous repudiable ring signature scheme from [7] with the construction of this paper. For notation, we denote the size of the ring R by n and the security parameter by λ.

	SignatureSize	Repudiable	RepudiationSize	Repudiation-Unforgeability
R-RS [7]	$n^2 \cdot \text{poly}(\lambda)$	✓	$n^2 \cdot \text{poly}(\lambda)$	×
Ours	$\log(n) \cdot \text{poly}(\lambda)$	✓	$\log(n) \cdot \text{poly}(\lambda)$	✓

a repudiation ξ for any signature σ with respect to any non-signer, and VerRepud is an algorithm which can verify whether ξ is a valid repudiation. The Repudiate and VerRepud algorithms are two non-interactive algorithms.

The repudiability for a ring signature scheme is a necessary property in some situations. For example, if we use a completely anonymous ring signature scheme in the blockchain, we will get a completely *Decentralized Anonymous Payment* (DAP) system, such as Monero [2,9]. But this system can be exploited by criminal activities [10], such as money laundering, payment of ransom for ransomware, online extortion, etc. Lin et al. [10] introduced the first *Decentralized Conditional Anonymous Payment* (DCAP) system to strike a balance between privacy protection and regulation. Their DCAP scheme is based on a *Conditional Anonymous Payments* (CAP) scheme. It looks like that if we use a repudiable ring signature scheme in the blockchain, we can also get a new DCAP system. This new DCAP system can be achieved by simply replacing the ring signature component of the existing DAP systems. This is our future work.

Although the repudiable ring signature scheme is very useful, the existing repudiable ring signature scheme still has some shortcomings. First, the security model of the repudiable ring signature scheme provided in the prior work [7] seems somewhat weak, in their paper they did not consider the unforgeability of the repudiation. Namely, the prior scheme may allow an adversary to forge a repudiation on behalf of some honest ring members. This is obviously unfair to the honest ring member, who may not be willing to generate this repudiation. Second, the size of signatures and repudiations of the prior scheme increases with the square of the number of ring members. This means as the number of ring members grows, the size of signatures and repudiations increases dramatically. So how to design a sub-linear repudiable ring signature scheme is also an urgent problem to be solved.

1.1. Our contributions

In this paper, we solve the existing problems mentioned above. In particular, we present a stronger security model for the repudiable ring signature scheme and construct a more efficient repudiable ring signature scheme that satisfies this stronger security.

In terms of security model, we propose the first formal definition of *repudiation-unforgeability*. *Repudiation-unforgeability* means that no one can produce a valid repudiation on behalf of any ring member, as long as he does not have the corresponding signing key. At first glance, this property seems to be included in the anonymity property,[2] it actually requires more in some aspects than the anonymity property. To illustrate this, let us consider the following specific setting. Suppose an adversary knows the identity of the real signer of a certain signature by some method (such as side-channel attacks, corrupt the signer, etc.). At this time, the anonymity of this signature no longer exists, but *repudiation-unforgeability* still guarantees that the adversary cannot generate a repudiation for other ring members. Although it is not clear the impact about the forgeability of the repudiation in practice, it is believed that the weaker ability the adversary has, the higher security the scheme guarantees, since this ability might be leveraged by the adversary to launch some potential attacks. We leave the impact about the forgeability of the repudiation in practice as an open problem.

Furthermore, Park et al. used the notion *adaptive anonymity against adversarially chosen keys* in [7]. Unfortunately, we find their construction cannot satisfy this property. We can give an attack for their scheme, namely there exist an adversary that can get who the actual signer is, the algorithm is given in the Appendix B. To rule out this attack, we need to limit the ability of the adversary slightly. Our modification is that, we do not allow the adversary to ask its *repudiation oracle* $\text{OR}(\cdot)$ for $(\cdot, m, \text{R}, \cdot)$ after the adversary gives the challenge information (i_0, i_1, m, R) to the experiment. Their repudiable ring signature scheme satisfies this modified anonymity, so does our scheme.[3]

In terms of construction, we give the first logarithmic-size repudiable ring signature scheme which does not rely on a trusted setup or the random oracle model. To do this, we use a *somewhere perfectly binding hash family* (SPB)[12] to compress the ring R, and then use the digest of R to calculate signatures and repudiations. Our repudiable ring signature scheme is also the first scheme which satisfies *repudiation-unforgeability*. To ensure this, we make evaluations of *verifiable random function* (VRF) as part of repudiation, and use the pseudorandomness of VRF to conceal the valid information.

1.2. Comparison to previous work

In this section, we compare our new scheme with the construction that was presented in [7]. To do this, we briefly review their repudiable ring signature construction and compare it to our work. For a more detailed exposition on their scheme, please refer to [7]. We summarize our comparisons in Table 1. We denote the size of the ring R by n and the security parameter by λ.

Construction in [7]. From a high level, the signature in [7] consists of every ring member's ZAP proofs which prove that some VRF values in the signature are correct. The repudiation for individual i consists of every ring member's ZAP proofs which prove that some VRF values in the signature are different from the values for party i's VRF evaluated at the message. Since the size of the witness for the membership proofs is linear in n, every signature and repudiation consist of n ZAP proofs, the signature and repudiation are size $n^2 \cdot \text{poly}(\lambda)$. Besides, they did not consider *repudiation-unforgeability* in their paper, i.e. their scheme may allow adversary to generate repudiation on behalf of some honest ring members.

Our Construction. It is clear that our repudiable ring signature construction is more efficient compared to the previous work. In particular, the signature and repudiation are size $\log(n) \cdot \text{poly}(\lambda)$. Furthermore, our repudiable ring signature scheme satisfies *repudiation-unforgeability*.

1.3. Related work

After the initial work of Rivest et al. [1], a number of works provided constructions under various computational hardness assumptions. The scheme of Dodis et al. [13] was the first to achieve sublinear size signatures in the ROM. Libert et al. [14] constructed a scheme with logarithmic size ring signature from DDH in the ROM. Recently, Backes et al. [15] provided the first standard model construction with signatures of size $\log(n)$.

Since the original proposal of ring signatures, various variant definitions also have been proposed. For example, linkable ring signatures

[2] Since if an adversary can forge a repudiation for a ring member, then he must not be the signer. Thus the anonymity naturally involves a part of repudiation-unforgeability.

[3] After we pointed out this problem, Park et al. also modified their anonymity definition in their updated version [11], their modified anonymity definition is the same as the anonymity definition given in our paper.

[16] allow identification of signatures that were produced by the same signer, without compromising the anonymity of the signer within the ring. Threshold ring signatures [17,18] can efficiently prove that a certain minimum number of users of a certain group must have actually collaborated to produce the signature, while hiding the precise membership of the subgroup. Proxy ring signatures [19–21] which allow proxy signer to sign message on behalf of the original signer provide anonymity. Identity-based ring signcryption [21,22] which can simplify key management procedures. Another notion called traceable ring signature [23] 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 [24,25] allow a signer to assign the power to deanonymize his signature to a specific publicly identified party. And recently, Park and Sealfon [7] proposed four new notions which are repudiable, unrepudiable, claimable and unclaimable ring signatures.

2. Preliminaries

Throughout the paper, let λ denote the security parameter and $\mathrm{negl}(\lambda)$ denote the negligible function. Denote by $y \leftarrow \mathscr{A}(x;r)$ the execution of algorithm that $\mathscr{A}$ output y, on input x and random coins r. Write $y \leftarrow \mathscr{A}(x)$, if the specific random coins used are not important. We denote by $y = \mathscr{A}(x)$, 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 $\{1, \cdots, n\}$.

In the following, we will briefly review some building blocks, which are *non-interactive witness-indistinguishable proof system* (NIWI), *verifiable random function* (VRF) and *somewhere perfectly binding hash function* (SPB). For the formal definitions of the used building blocks, see Appendix A.

2.1. Non interactive witness indistinguishable proof

Let $\mathscr{R}$ be an efficiently computable binary relation, where for $(x, w) \in \mathscr{R}$ we call x is a statement and w is a witness of x. Moreover, let $\mathscr{L}_{\mathscr{R}}$ denote the language consisting of all statements in $\mathscr{R}$, i.e. $\mathscr{L}_{\mathscr{R}} = \{x | \exists w : (x, w) \in \mathscr{R}\}$ A non interactive witness indistinguishable proof system for language $\mathscr{L}_{\mathscr{R}}$ is a pair of PPT algorithms (Prove, Verify), satisfying completeness, soundness and witness indistinguishability. More specifically, Prove is a probabilistic algorithm that takes as input a security parameter 1^λ, a statement x and a witness w, and return either a proof π or $\bot$; Verify is a probabilistic algorithm that takes as input a statement x and a proof π, outputs either 0 or 1. Just like [15], we require the size of the proof π satisfies $|\pi| = |C_x| \cdot \mathrm{poly}(\lambda)$, where C_x is a verification circuit for the statement x.

NIWI can be constructed from NIZK proofs derandomization assumptions [26] [27], from indistinguishability obfuscation and one-way permutations [28] and from bilinear pairings [29].

2.2. Verifiable random function

Let $a : \mathbb{N} \rightarrow \mathbb{N} \cup \{*\}$ and $b : \mathbb{N} \rightarrow \mathbb{N}$ be any two functions such that $a(\lambda)$, $b(\lambda)$ are both computable in time $\mathrm{poly}(\lambda)$, and they are both bounded by a polynomial in λ (expect when $a(\lambda)$ takes on the value $*$).[4] 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), and satisfies completeness, uniqueness and pseudorandomness. More specifically, Gen is a probabilistic algorithm that takes as input a security parameter 1^λ, and outputs a pair of keys (pk, sk); Eval is a deterministic algorithm that takes as input a secret key sk and $x \in \{0,1\}^{a(\lambda)}$, and outputs $y \in \{0,1\}^{b(\lambda)}$; Prove is a probabilistic algorithm that takes as input a secret key sk and $x \in \{0,1\}^{a(\lambda)}$, and outputs a proof π; and Verify is a probabilistic algorithm that 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. For simplicity, in this paper we assume that Eval takes inputs x of any length, i.e. $a(\lambda)$ takes the value of $*$.

The notion of VRF was introduced by Micali, Rabin and Vadhan [30]. Known constructions of VRFs are due to [30] based on strong RSA, [31] based on a strong version of the Diffie-Hellman assumption in bilinear groups, [32] based on the sum-free generalized DDH assumption, and [33] based on the bilinear Diffie-Hellman inversion assumption.

2.3. Somewhere perfectly binding hash function

The notion of somewhere perfectly binding hash function(SPB)[5] was introduced by [15], which can be used to create a short digest $h = \mathrm{H}_{\mathrm{hk}}(x)$ of some long input $x = (x[1], \cdots, x[n]) \in \Sigma^n$, where Σ is some alphabet. The hashing key $(\mathrm{hk}, \mathrm{shk}) \leftarrow \mathrm{Gen}(i)$ can be generated by providing a special 'binding index' i and this ensures that the hash $h = \mathrm{H}_{\mathrm{hk}}(x)$ is perfectly binding for the i'th symbol. In other words, even though h has many other preimages x' such that $\mathrm{H}_{\mathrm{hk}}(x') = h$, 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 allows us to prove that i'th symbol of x takes on some particular value $x[i] = u$ by providing a short opening π.

A somewhere perfectly binding hash family with private local opening SPB is given by a tuple of algorithms (Gen, Hash, Open, Verify), and satisfies correctness, somewhere perfectly binding and index hiding. More specifically, Gen is a probabilistic algorithm that 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 is a probabilistic algorithm that takes as input a hashing key hk and a database x and outputs a digest h; Open is a probabilistic algorithm that takes as input a hashing key hk, a private key shk, a database x and an index j and outputs a witness π; and Verify is a probabilistic algorithm that 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. Just like [15], 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 $\log(n) \cdot \mathrm{poly}(\lambda)$.

The notion of somewhere perfectly binding hash family with private local opening (SPB) was introduced by [15]. 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 [12] can be proofed to be SPB hash family.

3. Definitions of repudiable ring signatures

In this section, we provide definitions of repudiable ring signatures, which have a stronger security compared with [7]. Specifically, we add a new security requirement for repudiable ring signatures, which called *repudiation unforgeability*. Besides, we modify the definitions of *anonymity* and *repudiability* slightly compared with [7].

Definition 3.1. (*Repudiable ring signature*) A repudiable ring signature scheme is a tuple of PPT algorithms RRS = (Gen, Sign, Verify, Repudiate, VerRepud), that satisfies correctness, anonymity, unforgeability, repudiability and repudiation-unforgeability. The syntax of RRS follows:

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

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

[5] This is a stronger notion, compare with SSB in [34].

$$\Pr\left[\begin{array}{l|l} b=b' & (\mathrm{VK}_1,\mathrm{SK}_1),\cdots,(\mathrm{VK}_l,\mathrm{SK}_l)\leftarrow \mathrm{Gen}(1^\lambda) \\ \wedge i_0,i_1\in[l] & (m^*,\mathrm{R}^*,i_0,i_1,\mathfrak{s})\leftarrow \mathcal{A}_1^{\mathcal{O}_1}(1^\lambda,\tilde{\mathrm{R}}) \\ \wedge \mathrm{VK}_{i_0},\mathrm{VK}_{i_1}\in \mathrm{R}^* & b\leftarrow\{0,1\},\ \sigma\leftarrow \mathrm{Sign}(\mathrm{SK}_{i_b},m^*,\mathrm{R}^*) \\ \wedge \mathrm{VK}_{i_0},\mathrm{VK}_{i_1}\notin \mathcal{Q} & b'\leftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathfrak{s},\sigma) \end{array}\right] < \frac{1}{2}+\mathrm{negl}(\lambda) \qquad (1)$$

Fig. 1. Anonymity experiment.

$$\Pr\left[\begin{array}{l|l} b=1 & (\mathrm{VK}_1,\mathrm{SK}_1),\cdots,(\mathrm{VK}_l,\mathrm{SK}_l)\leftarrow \mathrm{Gen}(1^\lambda) \\ \wedge \mathrm{R}^*\subset\tilde{\mathrm{R}}\setminus\mathcal{Q}_{\mathrm{OC}} & (m^*,\mathrm{R}^*,\sigma)\leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda,\tilde{\mathrm{R}}) \\ \wedge(\cdot,m^*,\mathrm{R}^*)\notin\mathcal{Q}_{\mathrm{OS}} & b\leftarrow \mathrm{Verify}(m^*,\mathrm{R}^*,\sigma) \end{array}\right] < \mathrm{negl}(\lambda) \qquad (2)$$

Fig. 2. Unforgeability experiment.

$$\Pr\left[\begin{array}{l|l} & (\mathrm{VK},\mathrm{SK})\leftarrow \mathrm{Gen}(1^\lambda) \\ b=1\wedge b'=0 & (m^*,\mathrm{R}^*,\sigma)\leftarrow \mathcal{A}_1^{\mathcal{O}_1}(1^\lambda,\mathrm{VK}) \\ \wedge Q_1\cap\{(\cdot,m^*,\mathrm{R}^*)\}=\emptyset & \xi=\mathrm{Repudiate}(\mathrm{SK},m^*,\mathrm{R}^*,\sigma) \\ \wedge Q_1'\cap\{(\cdot,m^*,\mathrm{R}^*,\cdot)\}=\emptyset & b\leftarrow \mathrm{Verify}(m^*,\mathrm{R}^*,\sigma) \\ & b'\leftarrow \mathrm{VerRepud}(\mathrm{VK},m^*,\mathrm{R}^*,\sigma,\xi) \end{array}\right] < \mathrm{negl}(\lambda) \qquad (3)$$

$$\Pr\left[\begin{array}{l|l} & (\mathrm{VK}_1,\mathrm{SK}_1),\cdots,(\mathrm{VK}_l,\mathrm{SK}_l)\leftarrow \mathrm{Gen}(1^\lambda) \\ \mathrm{R}^*\cap\tilde{\mathrm{R}}\neq\emptyset & (m^*,\mathrm{R}^*,\sigma,\{\xi_{\mathrm{VK}}\}_{\mathrm{VK}\in\mathrm{R}^*\setminus\tilde{\mathrm{R}}})\leftarrow \mathcal{A}_2^{\mathcal{O}_2}(1^\lambda,\tilde{\mathrm{R}}) \\ b=1\wedge(\bigwedge_{\mathrm{VK}\in\mathrm{R}^*\setminus\tilde{\mathrm{R}}} b'_{\mathrm{VK}}=1) & b\leftarrow \mathrm{Verify}(m^*,\mathrm{R}^*,\sigma) \\ \wedge Q_2\cap\{(\cdot,m^*,\mathrm{R}^*)\}=\emptyset & \forall\ \mathrm{VK}\in\mathrm{R}^*\setminus\tilde{\mathrm{R}}, \\ & b'_{\mathrm{VK}}\leftarrow \mathrm{VerRepud}(\mathrm{VK},m^*,\mathrm{R}^*,\sigma,\xi) \end{array}\right] < \mathrm{negl}(\lambda) \qquad (4)$$

Fig. 3. Repudiability experiment.

$$\Pr\left[\begin{array}{l|l} \mathrm{VK}\in\mathrm{R}^* & (\mathrm{VK},\mathrm{SK})\leftarrow \mathrm{Gen}(1^\lambda) \\ \wedge b=1\wedge b'=1 & (m^*,\mathrm{R}^*,\sigma,\xi)\leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda,\mathrm{VK}) \\ \wedge Q\cap\{(\cdot,m^*,\mathrm{R}^*)\}=\emptyset & b\leftarrow \mathrm{Verify}(m^*,\mathrm{R}^*,\sigma) \\ \wedge Q'\cap\{(\cdot,m^*,\mathrm{R}^*,\cdot)\}=\emptyset & b'\leftarrow \mathrm{VerRepud}(\mathrm{VK},m^*,\mathrm{R}^*,\sigma,\xi) \end{array}\right] < \mathrm{negl}(\lambda) \qquad (5)$$

Fig. 4. Repudiation-unforgeability experiment.

$$\boxed{\begin{array}{c} (m,r,y_0^0,y_0^1,y_1^0,y_1^1,\mathrm{hk}_0,\mathrm{hk}_1,h_0,h_1)\in\mathcal{L}_1\Leftrightarrow \\ \exists\ \mathrm{VK},i,\eta,\tau^0,\tau^1,j\in\{0,1\}\text{ s.t.} \\ \mathrm{S.Verify}(\mathrm{hk}_j,h_j,i,\mathrm{VK},\eta)=1\wedge \\ \mathrm{V.Verify}(pk^0,(h_j,m;r),y_j^0,\tau^0)=1\wedge \\ \mathrm{V.Verify}(pk^1,(h_j,m;r),y_j^1,\tau^1)=1. \end{array}}$$

Fig. 5. $\mathscr{L}_1$ Language.

Sign(SK,m,R): takes as input a signing key SK, a message m and a ring $\mathrm{R}=(\mathrm{VK}_1,\cdots,\mathrm{VK}_n)$, and outputs a signature σ.
Verify(m,R,σ): takes as input a message m, a ring R 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 ring R 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 ring R, a signature σ and a repudiation ξ, and outputs either 0 or 1.

A repudiable ring signature scheme needs to satisfy five properties, expressed by Definition 3.2, 3.3, 3.4, 3.5, 3.6 below. In particular, *correctness, anonymity* and *unforgeability* are the properties that the usual ring signature scheme should hold, *repudiability* and *repudiation-*

$$\boxed{\begin{array}{c}(\mathrm{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0, \tilde{h}_1, z_0, z_1) \in \mathcal{L}_2 \Leftrightarrow \\ \exists\ y_0, y_1, \tau_{00}, \tau_{01}, \tau_{10}, \tau_{11} \text{ s.t.} \\ \mathrm{V.Verify}(pk^1, (\tilde{h}_0, m; r), y_0, \tau_{00}) = 1 \wedge \\ \mathrm{V.Verify}(pk^1, y_0, z_0, \tau_{01}) = 1 \wedge \\ \mathrm{V.Verify}(pk^1, (\tilde{h}_1, m; r), y_1, \tau_{10}) = 1 \wedge \\ \mathrm{V.Verify}(pk^1, y_1, z_1, \tau_{11}) = 1 \wedge \\ [(y_0 \neq y_0^1) \wedge (y_1 \neq y_1^1)] = 1.\end{array}}$$

Fig. 6. $\mathscr{L}_2$ Language.

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

Fig. A.7. Pseudorandomness experiment.

$$\Pr\left[\mathrm{Verify}(\mathrm{hk}, h, i, x[i], \pi) = 1 \;\middle|\; \begin{array}{l}(\mathrm{hk}, \mathrm{shk}) \leftarrow \mathrm{Gen}(1^\lambda, n, i) \\ h = \mathrm{Hash}(\mathrm{hk}, x) \\ \pi \leftarrow \mathrm{Open}(\mathrm{hk}, \mathrm{shk}, x, i)\end{array} \right] = 1. \tag{A.2}$$

Fig. A8. Correctness experiment.

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l}(n, i_0, i_1) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\mathrm{hk}, \mathrm{shk}) \leftarrow \mathrm{Gen}(1^\lambda, n, i_b), b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}_2(1^\lambda, (\mathrm{hk}, \mathrm{shk}))\end{array} \right] \leq \frac{1}{2} + \mathrm{negl}(\lambda). \tag{A.3}$$

Fig. A9. Index-hiding experiment.

unforgeability are the special properties that only repudiable ring signature schemes need to hold. Informally, *unforgeability* requires 'no one can forge a ring signature', *repudiability* requires 'non-signer can repudiate' and 'signer cannot repudiate', and *repudiation-unforgeability* requires 'no one can forge a valid repudiation for others'. In particular, the *unforgeability* guarantees that a ring signature which can be verified must be generated by some ring member; the *repudiability* guarantees that for a ring signature that can be verified, members of the ring (except the signer) are allowed to prove to outside that they are not the signer by generating a repudiation when necessary;[6] the *repudiation-unforgeability* guarantees that the repudiation must generated by himself, not forged by others.

To give the formal definitions of these properties, we need to introduce three oracles first:

- `Corruption oracle`: For a RRS scheme, the oracle $\mathrm{OC}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}$ is defined to take as input an index $i \in [l]$, and outputs a signing key SK_i.
- `Signing oracle`: For a RRS scheme, the signing oracle $\mathrm{OS}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}$ is defined to take as input an index $i \in [l]$, a message m and a ring R, and outputs a signature $\mathrm{Sign}(\mathrm{SK}_i, m, \mathrm{R})$.
- `Repudiation oracle`: For a RRS scheme, the oracle $\mathrm{OR}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}$ is defined to take as input an index $i \in [l]$, a message m, a ring R and a signature σ, and outputs a repudiation $\mathrm{Repudiate}(\mathrm{SK}_i, m, \mathrm{R}, \sigma)$.

Remark 1. We allow the ring R may contain some maliciously chosen verification keys in above oracles, i.e., there might be some verification keys $\widetilde{\mathrm{VK}_{j_1}}, \cdots, \widetilde{\mathrm{VK}_{j_k}} \in \mathrm{R}$ s.t. $\widetilde{\mathrm{VK}_{j_1}}, \cdots, \widetilde{\mathrm{VK}_{j_k}} \notin \{\mathrm{VK}_1, \cdots, \mathrm{VK}_l\}$, where VK_i is the verification key corresponding to SK_i.

In this paper, we use the same definition of correctness as in [7], which requires that verification can be successful for honestly generated signatures with respect to rings containing maliciously chosen verification keys.

Definition 3.2. (*Correctness*) We say that a RRS scheme satisfies correctness, if for every security parameter λ every PPT adversary $\mathscr{A}$, and every message m, we have the following formula holds.

$$\Pr\left[\mathrm{Verify}(\mathrm{m}, \mathrm{R}, \sigma) = 1 \;\middle|\; \begin{array}{l}(\mathrm{VK}, \mathrm{SK}) \leftarrow \mathrm{Gen}(1^\lambda) \\ \left(m, \widetilde{\mathrm{R}}\right) \leftarrow \mathscr{A}(1^\lambda, \mathrm{VK}) \\ \mathrm{R} = \mathrm{VK} \cup \widetilde{\mathrm{R}} \\ \sigma = \mathrm{Sign}(\mathrm{SK}, m, \mathrm{R})\end{array} \right] = 1.$$

In this paper, we refer to adaptive anonymity against adversarially chosen keys, which is slightly different from the previous one in [7]. In

[6] For example, a criminal used a ring signature scheme to create an anonymous threat letter. Then other members in the ring can use their signing key to generate a repudiation to exclude suspicion.

```
$(\mathrm{VK}, \mathrm{SK}) \leftarrow \textbf{RRS.Gen}(1^\lambda)$ :
  1: Generate two VRF key pairs:
         $(pk^0, sk^0),\ (pk^1, sk^1) \leftarrow \mathrm{V.Gen}(1^\lambda)$.
  2: Output $\mathrm{VK} = (pk^0, pk^1)$, $\mathrm{SK} = (sk^0, sk^1, \mathrm{VK})$.[1]

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

$b \leftarrow \textbf{RRS.Verify}(m, \mathrm{R}, \sigma)$ :
  1: Parse $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$.
  2: Compute $h_0\prime = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R})$, $h_1\prime = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R})$.
  3: Output the result:
         $\mathrm{N.Verify}_{\mathcal{L}_1}((m, r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, h_0\prime, h_1\prime), \pi)$.

$\xi \leftarrow \textbf{RRS.Repudiate}(\mathrm{SK}, m, \mathrm{R}, \sigma)$ :
  1: Parse R as above, SK= $(sk^0, sk^1, \mathrm{VK})$ and signature $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, if $\mathrm{VK} \notin \mathrm{R}$ output $\perp$ and halt. Else compute
     $b = \mathrm{RRS.Verify}(m, \mathrm{R}, \sigma)$, if $b = 0$ output $\perp$ and halt.
  2: Compute $\tilde{h}_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R})$, $\tilde{h}_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R})$.
  3: Compute VRF evaluations:
         $y_0 = \mathrm{V.Eval}(sk^1, (\tilde{h}_0, m; r)),\ y_1 = \mathrm{V.Eval}(sk^1, (\tilde{h}_1, m; r))$.
     If $y_0 = y_0^1$ or $y_1 = y_1^1$, output $\perp$ and halt.
  4: Compute $z_0 = \mathrm{V.Eval}(sk^1, y_0)$, $z_1 = \mathrm{V.Eval}(sk^1, y_1)$, and then compute VRF proofs
         $\tau_{00} \leftarrow \mathrm{V.Prove}(sk^1, (\tilde{h}_0, m; r))$, $\tau_{10} \leftarrow \mathrm{V.Prove}(sk^1, (\tilde{h}_1, m; r))$, $\tau_{01} \leftarrow \mathrm{V.Prove}(sk^1, y_0)$, $\tau_{11} \leftarrow \mathrm{V.Prove}(sk^1, y_1)$.
  5: Set $x = (\mathrm{VK}, m, r, y_0^1, y_1^1, \tilde{h}_0, \tilde{h}_1, z_0, z_1)$, witness $w = (y_0, y_1, \tau_{00}, \tau_{01}, \tau_{10}, \tau_{11})$, and then compute the proof $\pi\prime \leftarrow \mathrm{N.Prove}_{\mathcal{L}_2}(x, w)$.
  6: The repudiation is $\xi = (z_0, z_1, \pi\prime)$.

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

Algorithm 1. RRS Protocol

Table C.2
$\mathscr{B}$' answers in the "unforgeability" experiment

1. If $\mathscr{A}$ makes a corruption query for i^*, then $\mathscr{B}$ outputs a random bit and aborts.
2. If $\mathscr{A}$ queries its signing oracle OS$(\cdot)$ on (i, m, R), where $i \neq i^*$, $\mathscr{B}$ faithfully answers:RRS.Sign$(\mathrm{SK}_i, m, \mathrm{R})$ to $\mathscr{A}$.If $\mathscr{A}$ makes a signing query for (i^*, m, R), then $\mathscr{B}$ 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, $\mathscr{B}$ generates these by invoking its VRF oracle.
3. If $\mathscr{A}$ queries its repudiation oracle OR$(\cdot)$ on $(j, m, \mathrm{R}, \sigma)$, $\mathscr{B}$ faithfully answers RRS. Repudiate$(\mathrm{SK}_j, m, \mathrm{R}, \sigma)$ to $\mathscr{A}$.

Table C3
$\mathscr{B}$' answers in the "non-singer can repudiate" experiment.

1.When $\mathscr{A}$ makes a signing query for $(\cdot, m, \mathrm{R})$, $\mathscr{B}$ 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, $\mathscr{B}$ generates these by invoking its VRF oracle.
2. When $\mathscr{A}$ makes a repudiation query for inputs $(\cdot, m, \mathrm{R}, \sigma)$, $\mathscr{B}$ runs the honest repudiating algorithm RRS.Repudiate with the following modification: in step 3 and step 4, instead of using sk to generate $y_0, y_1, \tau_{00}, \tau_{10}, z_0, z_1$, and τ_{01}, τ_{11}, $\mathscr{B}$ generates these by invoking its VRF oracle.

particular, we require the adversary cannot query its OR$(\cdot)$ oracle at $(\cdot, m^*, \mathrm{R}^*, \cdot)$, where m^* and R^* are the challenge message and ring in the following experiment. In fact, the R-RS scheme in [7] only satisfies our anonymity definition and does not satisfy the definition they proposed. We can give an attack for their construction, which is given in Appendix B.

To give a formal definition, we define a punctured signing oracle first. Let $\mathrm{OR}^{(m^*,\mathrm{R}^*)}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}$ be a signing oracle which outputs $\perp$ when it receives both the ring R^* and message m^* as inputs, and otherwise gives the same response as $\mathrm{OR}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}$. We refer to this oracle as "punctured at m^* and R^*".

Definition 3.3. (*Anonymity*) We say a RRS scheme satisfies adaptive anonymity against adversarially chosen keys, if for every security parameter λ, every $l = \mathrm{poly}(\lambda)$ and every PPT adversary $\mathscr{A}$, we have the formula (1) (Fig. 1) holds, where $\widetilde{\mathrm{R}} = \{\mathrm{VK}_1, \cdots, \mathrm{VK}_l\}$,$\mathscr{O}_1 = (\mathrm{OC}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OS}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OR}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)})$, $\mathscr{O}_2 = (\mathrm{OC}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OS}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OR}^{(m^*,\mathrm{R}^*)}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)})$ and $\mathscr{Q}$ is the set of queries to OC.

Remark 2. We allow the ring R^* chosen by the adversary $\mathscr{A}_1$ may contain maliciously chosen verification keys that were not generated by the challenger.

We refer to unforgeable with respect to insider corruption, which is the same as the definition used in [7].

Definition 3.4. (*Unforgeability*) We say a RRS scheme is unforgeable with respect to insider corruption, if for every security parameter λ, every $l = \mathrm{poly}(\lambda)$ and every PPT adversary $\mathscr{A}$, we have the formula (2) (Fig. 2) holds, where $\widetilde{\mathrm{R}} = \{\mathrm{VK}_1, \cdots, \mathrm{VK}_l\}$,$\mathscr{O} = (\mathrm{OC}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OS}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OR}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)})$,$\mathscr{Q}_{\mathrm{OC}}$ is the set of queries to OC and $\mathscr{Q}_{\mathrm{OS}}$ is the set of queries to OS.

Repudiability requires two conditions, which are *non-signer can repudiate* and *signer cannot repudiate*. Intuitively, *non-signer can repudiate* captures the requirement that for any signature, a non signer can produce a valid repudiation; *signer cannot repudiate* captures the requirement that the signer cannot produce a valid repudiation. This requirement was proposed by [7] first.

In our work, we use a slightly different variant of the repudiability property. We require the adversary cannot queries its OR oracle at $(\cdot, m^*, \mathrm{R}^*, \cdot)$ in the *non-signer can repudiate* experiment, where m^* and R^* are the challenge message and ring .

Definition 3.5. (*Repudiability*) We say a RRS scheme satisfies repudiability, if the following two conditions hold.

1.(Non-signer can repudiate) For every security parameter λ and every PPT adversary $\mathscr{A}_1$ we have the formula (3) (Fig.3) holds, where $\mathscr{O}_1 = (\mathrm{OS}_{\mathrm{SK}}, \mathrm{OR}_{\mathrm{VK}})$, Q_1 is the set of queries to OS and Q_1' is the set of queries to OR.

2.(Signer cannot repudiate) For every security parameter λ, every $l = \mathrm{poly}(\lambda)$ and every PPT adversary $\mathscr{A}_2$ we have the formula (4) (Fig. 3) holds, where $\widetilde{\mathrm{R}} = \{\mathrm{VK}_1, \cdots, \mathrm{VK}_l\}$, $\mathscr{O}_2 = (\mathrm{OS}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)}, \mathrm{OR}_{(\mathrm{SK}_1,\cdots,\mathrm{SK}_l)})$, Q_2 is the set of queries to OS.

Remark 3. Our definition of the repudiability property is slightly different from the previous one in [7]. We require the adversary cannot queries its OR oracle at $(\cdot, m^*, \mathrm{R}^*, \cdot)$ in the experiment of "non-signer can repudiate", where m^* and R^* are the challenge message and ring in the experiment. In fact, we find the R-RS scheme in [7] does not satisfy the definition they proposed or at least their proof is not correct(Lemma 4.12 in [7]). The signing key of the R-RS scheme in [7] is $\mathrm{SK} = (\overrightarrow{sk}_{\mathrm{VRF}}, \mathrm{VK})$ where $\overrightarrow{sk}_{\mathrm{VRF}} = (sk_1, \cdots, sk_4)$ are four VRF secret keys. The main idea of the proof of their "non-signer can repudiate" experiment is that, if there exists a PPT adversary $\mathscr{A}$ that breaks R-RS scheme in [7] (in the sense of non-signer can repudiate) with non-negligible advantage, then we have $y_3 = \mathrm{V.Eval}(sk_3, (\mathrm{R}^*, m^*, \varphi))$ or $y_4 = \mathrm{V.Eval}(sk_4, (\mathrm{R}^*, m^*, \varphi))$ with non-negligible probability, where y_3, y_4 are the part of signature. But in the experiment, the adversary can output a signature with $y_2 = \mathrm{V.Eval}(sk_2, (\mathrm{R}^*, m^*, \varphi))$, and the rest of signature generated honestly, this type of adversary also has a non-negligible advantage to attack R-RS scheme. Because in the experiment, repudiation $\xi = (\xi_1, \cdots, \xi_n)$ is generated by the challenger, the witness used to generate proof ξ is $w = (i^*, y_1', y_2', \perp, \perp, \tau_1', \tau_2', \perp, \perp, \gamma)$. But w is not a valid witness, thus there are non-negligible probability such that ξ cannot pass the verifying. But this type of adversary is not considered in [7].

Furthermore, we also need that no one can produce a valid repudiation on behalf of other people, as long as he does not have signing key. We call this requirement *repudiation unforgeability*, this is a new requirement we proposed.

Definition 3.6. (*Repudiation-Unforgeability*) We say that a RRS scheme satisfies repudiation unforgeability, if for every security parameter λ and every PPT adversary $\mathscr{A}$, we have the formula (5) (Fig. 4) holds, where $\mathscr{O} = (\mathrm{OS}_{\mathrm{SK}}, \mathrm{OR}_{\mathrm{SK}})$, Q is the set of queries to OS and Q' is the set of queries to OR.

4. Construction of repudiable ring signatures

In this section, we describe our repudiable ring signature scheme which satisfies a stronger definition, has logarithmic-size signature and repudiation, and is based on general assumptions. To give a formal description, we define the following two languages first.

Let $\mathscr{L}_1$ denote the language (Fig. 5): In other word, $(m, r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, h_0, h_1) \in \mathscr{L}_1$ iff either (hk_0, h_0) binds 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) binds to a key VK, y_1^0 and y_1^1 are evaluations of $(h_1, m; r)$ for sk^0 and sk^1. This language is used to produce signature.

Let $\mathscr{L}_2$ denote the language (Fig. 6): In other word, $(\mathrm{VK}, m, r, y_0^1, y_1^1, \widetilde{h}_0, \widetilde{h}_1, z_0, z_1) \in \mathscr{L}_2$ 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 $(\widetilde{h}_0, m; r)$ and $(\widetilde{h}_1, m; r)$ for sk^1, and $y_0 \neq y_0^1$, $y_1 \neq y_1^1$. This language is used to produce repudiation.

4.1. Formal Description of RRS

Let (S.Gen, S.Hash, S.Open, S.Verify) be a somewhere perfectly binding hash function with private local opening and (V.Gen, V.Eval, V. Prove, V.Verify) be a verifiable random function with input domain $\{0,1\}^*$, output range $\{0,1\}^{\alpha(\lambda)}$. Let $(\mathrm{N.Prove}_{\mathscr{L}_1}, \mathrm{N.Verify}_{\mathscr{L}_1})$ be a NIWI-proof system for the language $\mathscr{L}_1$ and $(\text{N.Prove}_{\mathscr{L}_2}, \text{N.Verify}_{\mathscr{L}_2})$ be a NIWI-proof system for the language $\mathscr{L}_2$. In the rest of this section, we use the following convention to parse a ring R, write $\text{R} = \{\text{VK}_1, \cdots, \text{VK}_n\}$, and use $\text{R}[k]$ denote k'th verification key VK_k in R. Our repudiable ring signature scheme RRS is given in Algorithm 1.

Note that the first three algorithms can be used as a general ring signature system alone.

4.2. *Efficiency*

We now 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 are $\text{poly}(\lambda)$ and independent of the ring-size n. Since SPB is efficient, we have hk_0, hk_1 are 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 $\mathscr{L}_1$ 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 are $\text{poly}(\lambda)$ 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 also $\log(n)\cdot\text{poly}(\lambda)$.

5. Security of RRS

In this section, we present the security of our scheme. In particular, we prove that our RRS scheme satisfies five conditions which has been described in Section 3.

5.1. *Correctness*

We first prove our RRS scheme satisfies correctness when NIWI has completeness, VRF has completeness and SPB has correctness.

Theorem 5.1. *The repudiable ring signature scheme* RRS *satisfies correctness, given that* NIWI *has completeness,* VRF *has completeness and* SPB *has correctness.*

Proof. Assume (VK, SK) is generated by $\text{RRS.Gen}(1^\lambda)$, 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 RRS.Sign(SK, m, R), where $\text{R} = (\text{VK}_1, \cdots \text{VK}_n)$ is a ring generated by adversary, and $\text{R}[i] = \text{VK}$. 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, 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 τ^0, τ^1 such that it holds

$\text{V.Verify}(pk^0, (h_0, m; r), y_0^0, \tau^0) = 1, \text{V.Verify}(pk^1, (h_0, m; r), y_0^1, \tau^1) = 1.$

Therefore, $(m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1) \in \mathscr{L}_1$ 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

$\text{N.Verify}_{\mathscr{L}_1}((m, r, y_0^0, y_0^1, y_1^0, y_1^1, \text{hk}_0, \text{hk}_1, h_0, h_1), \pi) = 1$, and consequently $\text{RRS.Verify}(m, \text{R}, \sigma) = 1$.□

For the following four theorems, we only give the proof sketch here. Please refer Appendix C for the formal proof.

5.2. *Anonymity*

We now turn to establish the anonymity for our RRS scheme.

Theorem 5.2. *The repudiable ring signature scheme* RRS *satisfies anonymity against adversarially chosen keys, given that* NIWI *has witness indistinguishability,* VRF *has pseudorandomness, and* SPB *has index hiding.*

Proof Sketch The main idea of the proof is that, first move the index of hk_1 from i_0 to i_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 $(\text{VK}_{i_1}, \text{ind}_1, \eta_1, \tau_1^0, \tau_1^1, 1)$, and by witness indistinguishability of NIWI, this signature also satisfies indistinguishability. Then, 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_{i_1}.

5.3. *Unforgeability*

We will turn to show that RRS is unforgeable.

Theorem 5.3. *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.*

Proof Sketch The main idea of the proof is that, if $\mathscr{A}$ can forge a valid signature, then by soundness of NIWI, there exists a witness $w = (\text{VK}, i, \eta, \tau^0, \tau^1, j)$ s.t. we have

$\text{S.Verify}(\text{hk}_j, h_j, i, \text{VK}, \eta) = 1$, $\text{V.Verify}(pk^0, (h_j, m; r), y_j^0, \tau^0) = 1$, $\text{V.Verify}(pk^1, (h_j, m; r), y_j^1, \tau^1) = 1$, where $\text{VK} = (pk^0, pk^1)$. Since SPB has somewhere perfectly binding, we have $\text{VK} = \text{R}[i]$. And by the completeness and uniqueness of the VRF, we have

$y_j^0 = \text{V.Eval}(pk^0, (h_j, m, r))$, $y_j^1 = \text{V.Eval}(pk^1, (h_j, m, r))$, and by this we can attack the pseudorandomness of VRF.

5.4. *Repudiability*

We will turn to show that our RRS is repudiable.

Theorem 5.4. *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.*

Proof Sketch 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 honest 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 $\mathscr{A}$ can produce a valid repudiation for himself, then by the soundness of NIWI, we can prove that the signature which $\mathscr{A}$ outputs is not generated by himself. And by this we can also attack the pseudorandomness of VRF.

5.5. *Repudiation unforgeability*

We now turn to show that our RRS scheme satisfies repudiation unforgeability.

Theorem 5.5. *The repudiable ring signature scheme* RRS *satisfies repudiation unforgeability, given that* NIWI *has soundness,* VRF *has completeness, uniqueness, and pseudorandomness.*

Proof Sketch The main idea of the proof is that, if $\mathscr{A}$ can forge a valid repudiation for other member, then by the soundness of NIWI, there must be y_0, τ_{00}, τ_{01}, subject to

$\text{V.Verify}(pk^1, (h_0, m; r), y_0, \tau_{00}) = 1$, $\text{V.Verify}(pk^1, y_0, z_0, \tau_{01}) = 1$.

By completeness and uniqueness of VRF, 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.

6. Conclusions

Repudiable ring signature is an extension of the concept of ring signature, which allows non-signer to repudiate a signature that was not produced by him. Our work focuses on improving the security model of the repudiable ring signature as well as saving the size of signature and repudiation. In particular, we propose a new requirement for the repudiable ring signature, which is repudiation unforgeability. In terms of design, we present a new scheme which beats the state-of-the-art. An interesting open question is whether the ability of forgery repudiation can be used to launch some attacks on the repudiable ring signature.

Credit Author Statement

Hao Lin: Conceptualization, Data curation, Formal analysis, Methodology, Software, Visualization, Writing orginal draft. Mingqiang Wang: Funding acquisition, Investigation, Projet administration, Resources, Supervision, Validation, Writing review and editing. We the undersigned declare that this manuscript entitled Repudiable Ring Signature: Stronger Security and Logarithmic Logarithmic-Size is original, has not been published before and is not currently being considered for publication elsewhere.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We under stand that the Corresponding Author is the sole contact for the Editorial process. He is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Key Research and Development Program of China Stem Cell and Translational Research (Grant No. 2018YFA0704702) and National Natural Science Foundation of China (Grant Nos. 61832012 and 61672019).

Appendix A. Formal Definitions of Some Building Blocks

Definition A.1. (*NIWI*) A non interactive witness indistinguishable proof system NIWI for the language $\mathscr{L}_{\mathscr{R}}$ 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 $(x, w) \in \mathscr{R}$, we have

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

where the probability is taken over the randomness of Prove and Verify algorithms.
Soundness: for every λ, every $x \notin \mathscr{L}_{\mathscr{R}}$ 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

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

we have

$$\left\{\text{Prove}\left(1^\lambda, x, w_0\right)\right\} \overset{c}{\approx} \left\{\text{Prove}\left(1^\lambda, x, w_1\right)\right\}.$$

Definition A.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 probabilistic.
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.

We require it satisfies the following properties.

Completeness: for every λ and every $x \in \{0,1\}^{a(\lambda)}$, we have the following formula holds, where the probability is taken over the randomness of Gen, Prove and Verify algorithms.

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

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 $i = 1$:

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

where the probability is taken over the randomness of Verify algorithm.
Pseudorandomness: for any PPT adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ we have formula (A.1) (Fig. A.7) holds, where oracle $\mathscr{O} = (\mathrm{Eval}(sk,\cdot), \mathrm{Prove}(sk,\cdot))$ and $\mathscr{Q}$ is the set of oracle queries made by $\mathscr{A}$.

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

Gen$(1^{\lambda}, n, i)$:[7] 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 formula (A.2) (Fig. A.8) holds, 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 the following formula holds, where the hash key hk is generated by $\mathrm{Gen}(1^{\lambda}, n, \mathrm{ind})$.

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

Index Hiding: for any PPT adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ we have formula (A.3) (Fig. A.9) holds. To simplify notion, 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.

Remark 4. 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.

Appendix B. An Attack on [7]

In [7], they claim their repudiable ring signature scheme satisfies adaptive anonymity against adversarially chosen keys they proposed. But we find it is wrong, their construction can only satisfies anonymity (Definition 3.3) we proposed. Here, we will give an attack on the anonymity of their construction.
Let R-RS be the repudiable ring signature proposed in [7], and let OR$(\cdot)$ be the repudiation oracle.
Adversary $\mathscr{A}$: given 1^{λ}, $\mathrm{VK}_1, \cdots, \mathrm{VK}_l$, and OR$(\cdot)$.

1. The adversary $\mathscr{A}$ chooses $(m, \mathrm{R}, j_0, j_1)$ at random, and gives it to the experiment.
2. The challenger gives $\mathscr{A}$ a signature σ, $\mathscr{A}$ parses signature $\sigma = ((\pi_1, \cdots \pi_n), (y_1, \cdots y_4), \varphi)$, then $\mathscr{A}$ chooses $y' \leftarrow \{0,1\}^l$, and generates new signature: $\sigma' = ((\pi_1, \cdots \pi_n), (y_1, \cdots y_3, y'), \varphi)$.
3. The adversary $\mathscr{A}$ gives the input $j_0, m, \mathrm{R}, \sigma'$ to its repudiation oracle OR$(\cdot)$, and then get a repudiation $\xi = (\xi_1, \cdots, \xi_n)$.
4. The adversary $\mathscr{A}$ runs the honest verification algorithm R-RS.VerRepud and outputs the verification result R-RS$(\mathrm{R}, \mathrm{VK}_{j_0}, \sigma', \xi)$ to the challenger.

We find that if the challenger's bit $b = 0$, then by the definition of R-RS.Repudiate, in this situation ξ cannot pass through the verification. When $b = 1$, since R-RS.Repudiate only use y_1, y_2, this change does not affect the generation of reputation, and ξ can pass through the verification. Therefore, $\mathscr{A}$ has non-negligible advantage in the experiment.

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

Appendix C. Security Proofs of RRS

Proof of Theorem 5.2.. Let $\mathscr{A}$ be a PPT adversary against the anonymity of RRS scheme. Assume that $\mathscr{A}$ makes at most $q = \text{poly}(\lambda)$ queries for any oracle. 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} in R, where (m, R, i_0, i_1) is the challenge query of $\mathscr{A}$. Now, consider the following hybrids:

$\mathscr{H}_1$: This is the real experiment with challenge bit $b = 0$.
$\mathscr{H}_2$: Same as $\mathscr{H}_1$, except replace
hk_1 by $(\text{hk}_1, \text{shk}_1) \leftarrow \text{S.Gen}(1^\lambda, |\text{R}|, \text{ind}_1)$.
$\mathscr{H}_3$: Same as $\mathscr{H}_2$, except replace
y_1^1 by $y_1^1 = \text{V.Eval}(sk_{i_1}^1, (h_1, m; r))$.
$\mathscr{H}_4$: Same as $\mathscr{H}_3$, except replace y_1^0 by
$y_1^0 = \text{V.Eval}(sk_{i_1}^0, (h_1, m; r))$.
$\mathscr{H}_5$: Same as $\mathscr{H}_4$, except that we compute witness by
$\eta \leftarrow \text{S.Open}(\text{hk}_1, \text{shk}_1, \text{R}, \text{ind}_1)$, $\tau^0 \leftarrow \text{V.Prove}(sk_{i_1}^0, (h_1, m; r))$, $\tau^1 \leftarrow \text{V.Prove}(sk_{i_1}^1, (h_1, m; r))$,
and then use the witness $w = (\text{VK}_{i_1}, \text{ind}_1, \eta, \tau^0, \tau^1, 1)$ to compute π.
$\mathscr{H}_6$: Same as $\mathscr{H}_5$, except that we compute y_0^0 and y_0^1 by $y_0^0, y_0^1 \leftarrow \{0,1\}^{\alpha(\lambda)}$.
$\mathscr{H}_7$: Same as $\mathscr{H}_6$, except replace hk_0 by
$(\text{hk}_0, \text{shk}_0) \leftarrow \text{S.Gen}(1^\lambda, |\text{R}|, \text{ind}_1)$.
$\mathscr{H}_8$: Same as $\mathscr{H}_7$, except replace y_0^0 and y_0^1 by
$y_0^0 = \text{V.Eval}(sk_{i_1}^0, (h_0, m; r))$, $y_0^1 = \text{V.Eval}(sk_{i_1}^1, (h_0, m; r))$.
$\mathscr{H}_9$: Same as $\mathscr{H}_8$, except that we compute witness by
$\eta \leftarrow \text{S.Open}(\text{hk}_0, \text{shk}_0, \text{R}, \text{ind}_1)$, $\tau^0 \leftarrow \text{V.Prove}(sk_{i_1}^0, (h_0, m; r))$, $\tau^1 \leftarrow \text{V.Prove}(sk_{i_1}^1, (h_0, m; r))$,
and use the witness $w = (\text{VK}_{i_1}, \text{ind}_1, \eta, \tau^0, \tau^1, 0)$ to compute π.
$\mathscr{H}_{10}$: Same as $\mathscr{H}_9$, 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 $b = 1$.

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

Claim 1 $\mathscr{H}_1$ and $\mathscr{H}_2$ are computationally indistinguishable, given that SPB is index hiding. More specifically, there exists a reduction $\mathscr{R}_1$ such that

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

We will provide an informal description of $\mathscr{R}_1$. The $\mathscr{R}_1$ simulates $\mathscr{H}_1$ faithfully, until $\mathscr{A}$ outputs a challenge query (m, R, i_0, i_1). Then $\mathscr{R}_1$ gives $(|\text{R}|, \text{ind}_0, \text{ind}_1)$ to the SPB Index-Hiding experiment and receives a hashing key hk^*. $\mathscr{R}_1$ continues the simulation of $\mathscr{H}_1$ faithfully, except that in the challenge signature it sets $\text{hk}_1 = \text{hk}^*$. In the end, $\mathscr{R}_1$ outputs the output of $\mathscr{A}$.

Clearly, if the challenge bit of the Index-Hiding experiment is 0 then $\mathscr{R}_1$ simulates $\mathscr{H}_1$ perfectly. And if the challenge bit of the Index-Hiding experiment is 1 then $\mathscr{R}_1$ simulates $\mathscr{H}_2$ perfectly. The claim follows.

Claim 2 We claim that $\mathscr{H}_2$ and $\mathscr{H}_3$ are computationally indistinguishable, given that the VRF is pseudorandom. More specifically, there exists a reduction $\mathscr{R}_2$ such that

$$q \cdot \text{Adv}_{\text{VRF}}\left(\mathscr{R}_2^{\mathscr{A}}\right) \geq \text{Adv}_{\mathscr{H}_2, \mathscr{H}_3}(\mathscr{A}).$$

The reduction $\mathscr{R}_2$ receives as input a public key pk^*. $\mathscr{R}_2$ simulates $\mathscr{H}_2$ faithfully, except for the following. Before the simulation starts, $\mathscr{R}_2$ chooses a random index i^* and sets $\text{VK}_{i^*} = (pk_{i^*}^0, pk^*)$, where $pk_{i^*}^0$ is generated as in $\mathscr{H}_2$ and pk^* is the input of $\mathscr{R}_2$. $\mathscr{R}_2$ continues the simulation of $\mathscr{H}_2$ until $\mathscr{A}$ announces (i_0, i_1, m, R). If it holds $i_1 \neq i^*$, $\mathscr{R}_2$ outputs $\perp$. Otherwise, $\mathscr{R}_2$ continues the simulation of $\mathscr{H}_2$ faithfully, except that instead of computing y_1^1 itself, $\mathscr{R}_2$ sends $(h_1, m; r)$ to the VRF pseudorandomness experiment, the experiment return a value y, and $\mathscr{R}_2$ uses y in the challenge signature. $\mathscr{R}_2$ continues the simulation and outputs whatever $\mathscr{A}$ outputs.

Clearly, if the challenge bit of the VRF experiment is 0 then $\mathscr{R}_2$ simulates $\mathscr{H}_2$ perfectly, and if the challenge bit of the VRF experiment is 1 then $\mathscr{R}_2$ simulates $\mathscr{H}_3$ perfectly. And since we require $\mathscr{A}$ can not query $\text{OR}(\cdot)$ with $(\cdot, m, \text{R}, \cdot)$, after $\mathscr{A}$ received a challenge signature, $\mathscr{A}$ cannot query the VRF at the challenge value. Therefore, $\mathscr{H}_2$ and $\mathscr{H}_3$ are computationally indistinguishable.

Claim 3 We claim $\mathscr{H}_3$ and $\mathscr{H}_4$ are computationally indistinguishable, given that the VRF is pseudorandom. More specifically, there exists a reduction $\mathscr{R}_3$ such that

$$q \cdot \text{Adv}_{\text{VRF}}\left(\mathscr{R}_3^{\mathscr{A}}\right) \geq \text{Adv}_{\mathscr{H}_3, \mathscr{H}_4}(\mathscr{A}).$$

The proof follows analogously to the proof of Claim 2.

Claim 4 We claim $\mathscr{H}_4$ and $\mathscr{H}_5$ are computationally indistinguishable, give that NIWI is computationally witness indistinguishable. More specifically, there exists a reduction $\mathscr{R}_4$ against the witness indistinguishability of NIWI such that

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

The reduction $\mathscr{R}_4$ simulates $\mathscr{H}_4$ faithfully, until the challenge signature is computed. Instead of computing the proof π itself, $\mathscr{R}_4$ 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 $w_0 = (\text{VK}_{i_0}, \text{ind}_0, \eta_0, \tau_0^0, \tau_0^1, 0)$ and $w_1 = (\text{VK}_{i_1}, \text{ind}_1, \eta_1, \tau_1^0, \tau_1^1, 1)$ to the NIWI witness indistinguishability experiment. The experiment returns a proof π^*, and $\mathscr{R}_4$ use the proof π^* in the challenge signature. $\mathscr{R}_4$ continues the simulation of $\mathscr{H}_4$

faithfully and outputs whatever the simulated $\mathscr{A}$ outputs.

Clearly, if the challenge bit of the witness indistinguishability experiment is 0, then $\mathscr{R}_4$ simulates $\mathscr{H}_4$ perfectly. On the other hand, if the challenge bit is 1, then $\mathscr{R}_4$ simulates $\mathscr{H}_5$ perfectly. Thus, the claim follows.

Claim 5 $\mathscr{H}_5$ and $\mathscr{H}_6$ are computationally indistinguishable, given that the VRF is pseudorandom. The proof follows analogously to the proof of Claim 2.

Claim 6 $\mathscr{H}_6$ and $\mathscr{H}_7$ are computationally indistinguishable, given that SPB is index hiding. The proof follows analogously to the proof of Claim 1.

Claim 7 $\mathscr{H}_7$ and $\mathscr{H}_8$ are computationally indistinguishable, given that the VRF is pseudorandom. The proof follows analogously to the proof of Claim 2.

Claim 8 $\mathscr{H}_8$ and $\mathscr{H}_9$ are computationally indistinguishable, give that NIWI is computationally witness indistinguishable. The proof follows analogously to the proof of Claim 4.

Claim 9 $\mathscr{H}_9$ and $\mathscr{H}_{10}$ are computationally indistinguishable, given that VRF is pseudorandom. The proof follows analogously to the proof of Claim 2.□

Proof of Theorem 5.3.. Assume there exists a PPT adversary $\mathscr{A}$ that breaks the unforgeability experiment with non-negligible probability. Then we will construct an adversary $\mathscr{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability.

Let pk^* be the public key obtained from VRF challenger. $\mathscr{B}$ first runs RRS.Gen(1^λ) to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\mathrm{VK}_{i^*} = (pk^*, pk^1_{i^*})$. Then $\mathscr{B}$ simulates unforgeability experiment faithfully and answers $\mathscr{A}$'s oracle queries as described in Table C.2[8].

Let $(m^*, \mathrm{R}^*, \sigma)$ be the output of $\mathscr{A}$. Then $\mathscr{B}$ computes $t = \mathrm{RRS.Verify}(m^*, \mathrm{R}^*, \sigma)$. If $t = 0$, $\mathscr{B}$ outputs a random bit and aborts. Else $\mathscr{B}$ parses the signature as $\sigma = (r, y^0_0, y^1_0, y^0_1, y^1_1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, and computes $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R}^*)$, $h_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R}^*)$. Then $\mathscr{B}$ chooses a random bit $k \leftarrow \{0,1\}$ and submits $(h_k, m^*; r)$ to the VRF challenger and then receives response y. If $y = y^0_k$, $\mathscr{B}$ outputs 0. Otherwise, $\mathscr{B}$ outputs a random bit.

It remains to show that $\mathscr{B}$ 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 $\mathscr{A}$ is unaffected by $\mathscr{B}$'s choice of i^*, until the point at which $\mathscr{A}$ submits an oracle query to oracle OC for input i^*. Since i^* is chosen at random by $\mathscr{B}$, it follows that with probability at least $\frac{1}{l}$ the adversary $\mathscr{A}$ does not trigger this abort.

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

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

Thus, by the soundness of NIWI, there exists a witness $w = (\mathrm{VK}, i, \eta, \tau^0, \tau^1, j)$, s.t. it holds

$\mathrm{S.Verify}(\mathrm{hk}_j, h_j, i, \mathrm{VK}, \eta) = 1$, $\mathrm{V.Verify}(pk^0, (h_j, m^*; r), y^0_j, \tau^0) = 1$, $\mathrm{V.Verify}(pk^1, (h_j, m^*; r), y^1_j, \tau^1) = 1$.

Since SPB has somewhere perfectly binding, we have $\mathrm{VK} = \mathrm{R}^*[i]$, and by the completeness and uniqueness of the VRF, we have

$y^0_j = \mathrm{V.Eval}(sk^0, (h_j, m^*; r))$, $y^1_j = \mathrm{V.Eval}(sk^1, (h_j, m^*; r))$.

When $i = i^*$ and $j = k$ happened, we have $y^0_k = \mathrm{V.Eval}(sk^*, (h_k, m^*; r))$. In this case, if the VRF challenger's bit $b = 0$, then we have $y = y^0_k$. Recall that this is the trigger condition for $\mathscr{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathscr{B}$, $\mathscr{B}$ 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. Since i^* is chosen at random by $\mathscr{B}$, it follows that with probability at least $\frac{1}{l}$, $i = i^*$ occurs. And since k is chosen at random by $\mathscr{B}$, it follows that with probability at least $\frac{1}{2}$, $j = k$ occurs.

Furthermore, since $\mathscr{A}$ cannot query its signing oracle $\mathrm{OS}(\cdot)$ on $(\cdot, m^*, \mathrm{R}^*)$, $\mathscr{B}$ has not previously made an oracle query on the VRF challenge message $(h_k, m^*; r)$ during the query phase.

All together, we can conclude that

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

This concludes the proof.□

Proof of Theorem 5.4.. We will prove each of the desired security properties in turn. *Non-signer can repudiate* Assume there exists a PPT adversary $\mathscr{A}$ that breaks our RRS scheme (in the non-singer can repudiate experiment) with non-negligible probability. Then we will construct an adversary $\mathscr{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability.

Let pk^* be the public key obtained from VRF challenger. $\mathscr{B}$ first runs RRS.Gen(1^λ) to generate a pair of keys, and set $\mathrm{VK} = (pk^0, pk^*)$. Then $\mathscr{B}$ simulates "non-singer can repudiate" experiment faithfully and answers $\mathscr{A}$'s oracle queries as described in Table C.3.

Let $(m^*, \mathrm{R}^*, \sigma)$ be the output of $\mathscr{A}$. Then $\mathscr{B}$ computes $t = \mathrm{RRS.Verify}(m^*, \mathrm{R}^*, \sigma)$. If $t = 0$, $\mathscr{B}$ outputs a random bit and aborts. Else $\mathscr{B}$ parses signature as $\sigma = (r, y^0_0, y^1_0, y^0_1, y^1_1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, and computes SPB hash value $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R}^*)$, $h_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R}^*)$. Then $\mathscr{B}$ chooses a random bit $k \leftarrow \{0,1\}$ and submits $(h_k, m^*; r)$ to the VRF challenger and then receive responses y. If $y = y^1_k$, $\mathscr{B}$ outputs 0. Otherwise, $\mathscr{B}$ outputs a random bit.

It remains to show that $\mathscr{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. Assume $\mathscr{A}$ wins the experiment, then we have VerRepud will reject an honestly generated repudiation ξ which generated by VK, i.e. $\mathrm{RRS.VerRepud}(\mathrm{VK}, m^*, \mathrm{R}^*, \sigma, \xi) = 0$, where $\xi \leftarrow \mathrm{RRS.Repudiate}(\mathrm{SK}, m^*, \mathrm{R}^*, \sigma)$. Since ξ is honestly generated with respect to VK, and by the definition of RRS.Repudiate and RRS.VerRepud, we have there exist $j \in \{0,1\}$ s.t. $y_j = y^1_j$, where $y_j = \mathrm{V.Eval}(sk^*, (\widetilde{h}_j, m^*; r))$. Since if the above formula are not true, then by the completeness of the NIWI and the completeness of the VRF, we have $\mathrm{RRS.VerRepud}(\mathrm{VK}, m^*, \mathrm{R}^*, \sigma, \xi) = 1$.

When $j = k$ happens, we have

$y^1_k = \mathrm{V.Eval}(sk^*, (h_k, m^*; r))$.

In this case, if the VRF challenger's bit $b = 0$, then we have $y = y^1_k$. Recall that this is the trigger condition for $\mathscr{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathscr{B}$, $\mathscr{B}$ outputs a random bit with overwhelm probability. Since k is chosen at random by $\mathscr{B}$, it follows that with probability $\frac{1}{2}$, $j = k$ occurs.

[8] Since sk is not used by RRS.Repudiate, $\mathscr{B}$ does not need to invoke the VRF oracle when $\mathscr{A}$ queries its repudiation oracle $\mathrm{OR}(\cdot)$.

Furthermore, since $\mathscr{A}$ cannot query its signing oracle $\mathrm{OS}(\cdot)$ on $(\cdot, m^*, \mathrm{R}^*)$ and cannot query its repudiation oracle on $(\cdot, m^*, \mathrm{R}^*, \cdot)$, $\mathscr{B}$ has not previously made an oracle query on the VRF challenge message $(h_k, m^*; r)$ during the query phase.

Besides, when $\mathscr{A}$ does not win the experiment, by the soundness of NIWI, we have $y \neq y_k^1$. In this case, $\mathscr{B}$ outputs a random bit.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathscr{B}) \geq \frac{1}{8} \cdot \mathrm{Adv}(\mathscr{A}).$$

This concludes the proof of non-signer can repudiate.

Signer cannot repudiate Assume there exists a PPT adversary $\mathscr{A}$ that breaks our RRS scheme (in the signer cannot repudiate experiment) with non-negligible probability. Then we will construct an adversary $\mathscr{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability.

Let pk^* be the public key obtained from the VRF challenger. $\mathscr{B}$ first runs $\mathrm{RRS.Gen}(1^\lambda)$ to generate l pairs of keys, then chooses a random index $i^* \in [l]$, and set $\mathrm{VK}_{i^*} = (pk^*, pk_{i^*}^1)$. Then $\mathscr{B}$ simulates "signer cannot repudiate" experiment faithfully and answers $\mathscr{A}$'s signing oracle $\mathrm{OS}(\cdot)$ and repudiation oracle $\mathrm{OR}(\cdot)$ queries in the same way as in the "unforgeability" experiment (Table C.2).

Let $(m^*, \mathrm{R}^*, \sigma, \{\xi_i\}_{\mathrm{VK}_i \in \mathrm{R}^* \setminus \widetilde{\mathrm{R}}})$ be the output of $\mathscr{A}$. $\mathscr{B}$ runs the verifying algorithm $t_0 = \mathrm{RRS.Verify}(m^*, \mathrm{R}^*, \sigma)$ and $t_i = \mathrm{RRS.VerRepud}(\mathrm{VK}_i, m^*, \mathrm{R}^*, \sigma, \xi_i)$ for all $\mathrm{VK}_i \in \mathrm{R}^* \setminus \widetilde{\mathrm{R}}$. If there exists a $t_i = 0$, $\mathscr{B}$ outputs a random bit and aborts. Else $\mathscr{B}$ parses $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$, and computes $h_0 = \mathrm{S.Hash}(\mathrm{hk}_0, \mathrm{R}^*)$, $h_1 = \mathrm{S.Hash}(\mathrm{hk}_1, \mathrm{R}^*)$. Then $\mathscr{B}$ chooses a random bit $k \leftarrow \{0, 1\}$ and submits $(h_k, m^*; r)$ to the VRF challenger and then receives response y. If $y = y_k^0$, $\mathscr{B}$ outputs 0. Otherwise, $\mathscr{B}$ outputs a random bit.

It remains to show that adversary $\mathscr{B}$ 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 $\mathscr{A}$ is unaffected by $\mathscr{B}$'s choice of i^*. Assume that $\mathscr{A}$ wins the experiment, then σ is a valid signature of m^* with respect to R^*. By soundness of NIWI, there exists a witness $w = (\mathrm{VK}, i, \eta, \tau^0, \tau^1, j)$, s.t. it holds $\mathrm{S.Verify}(\mathrm{hk}_j, h_j, i, \mathrm{VK}, \eta) = 1$,

$\mathrm{V.Verify}(pk^0, (h_j, m^*; r), y_j^0, \tau^0) = 1$, $\mathrm{V.Verify}(pk^1, (h_j, m^*; r), y_j^1, \tau^1) = 1$, where $\mathrm{VK} = (pk^0, pk^1)$. By somewhere perfectly binding of SPB, we have $\mathrm{VK} = \mathrm{R}^*[i]$. By the completeness and uniqueness of the VRF, we have $y_j^0 = \mathrm{V.Eval}(sk^0, (h_j, m^*; r))$ and $y_j^1 = \mathrm{V.Eval}(sk^1, (h_j, m^*; r))$.

We now show that we have $\mathrm{VK} \subset \widetilde{\mathrm{R}}$, i.e. VK is generated by $\mathscr{B}$. Since if $\mathrm{VK} \subset \mathrm{R}^* \setminus \widetilde{\mathrm{R}}$, then ξ_i is a valid repudiation with respect to VK. By soundness of NIWI, there exists a witness $w' = (y_0, y_1, \tau_{00}, \tau_{01}, \tau_{10}, \tau_{11})$, s.t. it holds

$\mathrm{V.Verify}(pk^1, (h_0, m^*; r), y_0, \tau_{00}) = 1$, $\mathrm{V.Verify}(pk^1, (h_1, m^*; r), y_1, \tau_{10}) = 1$.

By completeness and uniqueness of VRF, we have $y_0 = \mathrm{V.Eval}(pk^1, (h_0, m^*; r))$ and $y_1 = \mathrm{V.Eval}(pk^1, (h_1, m^*; r))$. Therefore, by the uniqueness of VRF, we have

$(y_0 = y_0^1) \vee (y_1 = y_1^1)$ with overwhelm probability. This is contradictory to the soundness of NIWI.

When $\mathrm{VK} = \mathrm{VK}_{i^*}$ 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 $\mathscr{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathscr{B}$, $\mathscr{B}$ outputs a random bit with overwhelm probability. Since i^* is chosen at random by $\mathscr{B}$, it follows that with probability at least $\frac{1}{l}$, $\mathrm{VK} = \mathrm{VK}_{i^*}$ occurs. And since k is chosen at random by $\mathscr{B}$, it follows that with probability at least $\frac{1}{2}$, $j = k$ occurs.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathscr{B}) \geq \frac{1}{8l} \cdot \mathrm{Adv}(\mathscr{A}).$$

This concludes the proof of signer cannot repudiate.□

Proof of Theorem 5.5.. Assume there exists a PPT adversary $\mathscr{A}$ that breaks the repudiation unforgeability experiment with non-negligible probability. Then we will construct an adversary $\mathscr{B}$ that breaks the pseudorandomness of the underlying VRF scheme with non-negligible probability.

Let pk^* be the public key obtained from VRF challenger. $\mathscr{B}$ first runs $\mathrm{RRS.Gen}(1^\lambda)$ to generate a pair of keys, and set $\mathrm{VK} = (pk^0, pk^*)$. Then $\mathscr{B}$ simulates "repudiation unforgeability" experiment faithfully and answers $\mathscr{A}$'s signing oracle $\mathrm{OS}(\cdot)$ and repudiation oracle queries in the same way as in the "non-singer can repudiate" experiment (Table C.3).

Let $(m^*, \mathrm{R}^*, \sigma, \xi)$ be the output of $\mathscr{A}$. $\mathscr{B}$ computes

$b = \mathrm{RRS.Verify}(m^*, \mathrm{R}^*, \sigma)$, $b' = \mathrm{RRS.VerRepud}(\mathrm{VK}, m^*, \mathrm{R}^*, \sigma, \xi)$.

If $b = 0$ or $b' = 0$, $\mathscr{B}$ outputs a random bit and aborts. Else $\mathscr{B}$ parses $\sigma = (r, y_0^0, y_0^1, y_1^0, y_1^1, \mathrm{hk}_0, \mathrm{hk}_1, \pi)$ and $\xi = (z_0, z_1, \pi')$, and computes $h_0 = \mathrm{SPB.Hash}(\mathrm{hk}_0, \mathrm{R}^*)$. $\mathscr{B}$ queries its oracle $\mathrm{Eval}(\cdot)$ on $(h_0, m^*; r)$ and get x, then $\mathscr{B}$ submits x to the VRF challenger and then receives a response y. If $y = z_0$, $\mathscr{B}$ outputs 0. Otherwise, $\mathscr{B}$ outputs a random bit.

It remains to show that adversary $\mathscr{B}$ has non-negligible advantage to attack the pseudorandomness of VRF. Assume $\mathscr{A}$ wins the experiment, then ξ will be a valid repudiation of VK respect to σ, i.e.

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

Thus, 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

$y_0 = \mathrm{V.Eval}(sk, (h_0, m^*; r))$, $z_0 = \mathrm{V.Eval}(sk, y_0)$.

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 $\mathscr{B}$ to output 0. If the VRF challenger's bit $b = 1$, then y is truly random strings. Thus, by the definition of $\mathscr{B}$, $\mathscr{B}$ outputs a random bit with overwhelm probability.

Furthermore, since $\mathscr{A}$ cannot query its signing oracle $\mathrm{OS}(\cdot)$ on $(\cdot, m^*, \mathrm{R}^*)$ and cannot query its repudiation oracle on $(\cdot, m^*, \mathrm{R}^*, \cdot)$, $\mathscr{B}$ has not previously made an oracle query on the VRF challenge message x during the query phase.

All together, we can conclude that

$$\mathrm{Adv}_{\mathrm{VRF}}(\mathscr{B}) \geq \frac{1}{4} \cdot \mathrm{Adv}_{\mathrm{Reun1}}(\mathscr{A}).$$

This conclude the proof.□

References

[1] R.L. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in: C. Boyd (Ed.), ASIACRYPT 2001, Lecture Notes in Computer Science, 2248, Springer, 2001, pp. 552–565.
[2] S. Noether, Ring signature confidential transactions for monero, IACR Cryptol. 2015 (2015) 1098.
[3] C. Schnorr, Efficient identification and signatures for smart cards, in: G. Brassard (Ed.), CRYPTO 1989, Lecture Notes in Computer Science, 435, Springer, 1989, pp. 239–252.
[4] Y. Komano, K. Ohta, A. Shimbo, S. Kawamura, Toward the fair anonymous signatures: deniable ring signatures, in: D. Pointcheval (Ed.), Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings, Lecture Notes in Computer Science, 3860, Springer, 2006, pp. 174–191.
[5] S. Zeng, S. Jiang, Z. Qin, A new conditionally anonymous ring signature, in: B. Fu, D. Du (Eds.), Computing and Combinatorics - 17th Annual International Conference, COCOON 2011, Dallas, TX, USA, August 14-16, 2011. Proceedings, Lecture Notes in Computer Science, 6842, Springer, 2011, pp. 479–491.
[6] W. Gao, L. Chen, Y. Hu, C.J.P. Newton, B. Wang, J. Chen, Lattice-based deniable ring signatures, Int. J. Inf. Secur. 18 (3) (2019) 355–370.
[7] S. Park, A. Sealfon, It Wasn't Me! - repudiability and claimability of ring signatures, , in: A. Boldyreva, D. Micciancio (Eds.), CRYPTO 2019, Lecture Notes in Computer Science, 11694, , Springer, 2019, pp. 159–190.
[8] H. Jia, C. Tang, Cryptanalysis of a non-interactive deniable ring signature scheme, Int. J. Inf. Secur. 20 (1) (2021) 103–112.
[9] C. Lin, D. He, H. Zhang, L. Shao, X. Huang, Privacy-enhancing decentralized anonymous credential in smart grids, Comput. Stand. Interfaces 75 (2021) 103505.
[10] C. Lin, D. He, X. Huang, M.K. Khan, K.R. Choo, DCAP: a secure and efficient decentralized conditional anonymous payment system based on blockchain, IEEE Trans. Inf. Forens.Secur. 15 (2020) 2440–2452.
[11] S. Park, A. Sealfon, It wasn't me! Repudiability and Unclaimability of Ring Signatures, IACR Cryptol. 2019 (2019) 135.
[12] T. Okamoto, K. Pietrzak, B. Waters, D. Wichs, New realizations of somewhere statistically binding hashing and positional accumulators, in: T. Iwata, J.H. Cheon (Eds.), ASIACRYPT 2015, Lecture Notes in Computer Science, 9452, Springer, 2015, pp. 121–145.
[13] Y. Dodis, A. Kiayias, A. Nicolosi, V. Shoup, Anonymous identification in ad hoc groups, in: C. Cachin, J. Camenisch (Eds.), EUROCRYPT 2004, Lecture Notes in Computer Science, 3027, Springer, 2004, pp. 609–626.
[14] B. Libert, T. Peters, C. Qian, Logarithmic-size ring signatures with tight security from the DDH assumption, in: J. López, J. Zhou, M. Soriano (Eds.), ESORICS 2018, Lecture Notes in Computer Science, 11099, Springer, 2018, pp. 288–308.
[15] M. Backes, N. Döttling, L. Hanzlik, K. Kluczniak, J. Schneider, Ring signatures: logarithmic-size, no setup - from standard assumptions, in: Y. Ishai, V. Rijmen (Eds.), EUROCRYPT 2019, Lecture Notes in Computer Science, 11478, Springer, 2019, pp. 281–311.
[16] J.K. Liu, V.K. Wei, D.S. Wong, Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract), in: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings, Lecture Notes in Computer Science, 3108, Springer, 2004, pp. 325–335.
[17] E. Bresson, J. Stern, M. Szydlo, Threshold ring signatures and applications to ad-hoc groups, in: M. Yung (Ed.), CRYPTO 2002, Lecture Notes in Computer Science, 2442, Springer, 2002, pp. 465–480.
[18] D.H. Duong, H.T.N. Tran, W. Susilo, L.V. Luyen, An efficient multivariate threshold ring signature scheme, Comput. Stand. Interfaces 74 (2021) 103489.
[19] J. Hu, J. Zhang, Cryptanalysis and improvement of a threshold proxy signature scheme, Comput. Stand. Interfaces 31 (1) (2009) 169–173.
[20] Y. Yu, C. Xu, X. Huang, Y. Mu, An efficient anonymous proxy signature scheme with provable security, Comput. Stand. Interfaces 31 (2) (2009) 348–353.
[21] M.R. Asaar, M. Salmasizadeh, W. Susilo, A short identity-based proxy ring signature scheme from RSA, Comput. Stand. Interfaces 38 (2015) 144–151.
[22] Z. Zhu, Y. Zhang, F. Wang, An efficient and provable secure identity-based ring signcryption scheme, Comput. Stand. Interfaces 31 (6) (2009) 1092–1097.
[23] E. Fujisaki, K. Suzuki, Traceable ring signature, in: T. Okamoto, X. Wang (Eds.), PKC 2007, Lecture Notes in Computer Science, 4450, Springer, 2007, pp. 181–200.
[24] S. Xu, M. Yung, Accountable ring signatures: a smart card approach, in: J. Quisquater, P. Paradinas, Y. Deswarte, A.A.E. Kalam (Eds.), Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France, IFIP, 153, Kluwer/Springer, 2004, pp. 271–286.
[25] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, C. Petit, Short accountable ring signatures based on DDH, in: G. Pernul, P.Y.A. Ryan, E.R. Weippl (Eds.), ESORICS 2015, Lecture Notes in Computer Science, 9326, Springer, 2015, pp. 243–265.
[26] C. Dwork, M. Naor, Zaps and their applications. FOCS 2000, IEEE Computer Society, 2000, pp. 283–293.
[27] B. Barak, S.J. Ong, S.P. Vadhan, Derandomization in cryptography, in: D. Boneh (Ed.), CRYPTO 2003, Lecture Notes in Computer Science, 2729, Springer, 2003, pp. 299–315.
[28] N. Bitansky, O. Paneth, ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation, in: Y. Dodis, J.B. Nielsen (Eds.), TCC 2015, Lecture Notes in Computer Science, 9015, Springer, 2015, pp. 401–427.
[29] J. Groth, R. Ostrovsky, A. Sahai, New techniques for noninteractive zero-knowledge, J. ACM (JACM) 59 (3) (2012) 11:1–11:35.
[30] S. Micali, M.O. Rabin, S.P. Vadhan, Verifiable random functions. FOCS 1999, IEEE Computer Society, 1999, pp. 120–130.
[31] A. Lysyanskaya, Unique signatures and verifiable random functions from the DH-DDH separation, in: M. Yung (Ed.), CRYPTO 2002, Lecture Notes in Computer Science, 2442, Springer, 2002, pp. 597–612.
[32] Y. Dodis, Efficient construction of (distributed) verifiable random functions, in: Y. Desmedt (Ed.), PKC 2003, Lecture Notes in Computer Science, 2567, Springer, 2003, pp. 1–17.
[33] Y. Dodis, A. Yampolskiy, A verifiable random function with short proofs and keys, in: S. Vaudenay (Ed.), PKC 2005, Lecture Notes in Computer Science, 3386, Springer, 2005, pp. 416–431.
[34] P. Hubácek, D. Wichs, On the communication complexity of secure function evaluation with long output, in: T. Roughgarden (Ed.), Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015, ACM, 2015, pp. 163–172.