International Journal of Information Security (2019) 18:355–370
https://doi.org/10.1007/s10207-018-0417-1

REGULAR CONTRIBUTION

Lattice-based deniable ring signatures

Wen Gao[1] · Liqun Chen[3] · Yupu Hu[2] · Christopher J. P. Newton[3] · Baocang Wang[2] · Jiangshan Chen[4]

Published online: 20 August 2018

Abstract
In cryptography, a ring signature is anonymous as it hides the signer's identity among other users. When generating the signature, the users are arranged as a ring. Compared with group signatures, a ring signature scheme needs no group manager or special setup and supports flexibility of group choice. However, the anonymity provided by ring signatures can be used to conceal a malicious signer and put other ring members under suspicion. At the other extreme, it does not allow the actual signer to prove their identity and gain recognition for their actions. A deniable ring signature is designed to overcome these disadvantages. It can initially protect the signer, but if necessary, it enables other ring members to deny their involvement, and allows the real signer to prove who made the signed action. Many real-world applications can benefit from such signatures. Inspired by the requirement for them to remain viable in the post-quantum age, this work proposes a new non-interactive deniable ring signature scheme based on lattice assumptions. Our scheme is proved to be anonymous, traceable and non-frameable under quantum attacks.

Keywords Lattice-based cryptography · Ring signature · Deniability · Deniable ring signature

1 Introduction

A digital signature allows a recipient to be confident about the source of information that they receive, but there is no anonymity, as the sender has to be identified. To give some degree of anonymity, a ring signature scheme [1] can be used. To send and validate information, a signer can take a number of people to form a ring and then generate a ring signature which allows the recipient to confirm that the information came from one of the ring members, but the actual individual cannot be identified. The ring members used to generate any given ring signature can vary.

Ring signatures have various applications which have been suggested already in previous works [1–3]. The original motivation was to leak secrets anonymously. For example, a high-ranking government officer can sign information with respect to the ring of all similarly high-ranking officers; the information can then be verified as coming from an important source without exposing the actual signer.

As described providing complete anonymity in a ring signature scheme may not always be desirable. Using a ring signature scheme can be open to abuse, where a malicious signer can use the anonymity to supply false information and put the other members of the ring under suspicion. On the other hand, when providing a piece of invaluable information, a ring signature scheme does not offer the real signer an opportunity to claim any credit in the future.

As the signer can always be traced by the group manager, group signatures [4–7], do not suffer from the issues outlined above. However, group signatures require a group manager and every signer has to join the group. Ring signatures are more flexible, as there is no centralized group manager or any requirement for coordination among the signers. Each signer has their individual private and public keys, and these are directly used for ring signatures. In addition, the recipient of the signed document knows the identity of the members of the ring and this can often add weight to the information in the document.

A deniable ring signature scheme, such as the one introduced by Komano et al. [8] in 2006, retains the flexibility

✉ Liqun Chen
liqun.chen@surrey.ac.uk

1 College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an, China

2 School of Communication Engineering, Xi'dian University, Xi'an, China

3 Department of Computer Science, University of Surrey, Guildford, UK

4 School of Mathematics and Statistics, Minnan Normal University, Zhangzhou Fujian, China

of a ring signature scheme, but can still protect members of the ring as it allows them to either confirm that they signed and sent the information, or to prove that they did not. Once issued the deniable ring signature can be validated and used and, certainly initially, the non-signing ring members have no reason to disavow their inclusion in the signature. Only later, if necessary, is the ability of any member of the ring to clarify their involvement in the signature used. Such deniable signatures have many applications, for example:

- *Anonymous online bidding at auction* Particularly for high value items bidders at auction often wish to remain anonymous. This scheme allows bidders to remain anonymous, but ensures that a successful bidder, having second thoughts, is unable to deny that they made the winning bid. The auction house knows where the bids are coming from and can, if necessary, find out the identity of the winning bidder.
- *Online criminal detection and reward system* The police rely on information from the public and, for serious crimes, often offer rewards for useful information. This scheme allows an informant to initially provide information anonymously and then, once the criminal is safely behind bars, to claim their reward while preventing someone else fraudulently doing so.
- *Anonymous disclosure system* Many organizations encourage workers to report illegal or abusive behavior by their colleagues. This scheme allows workers to do so anonymously, while guarding against malicious accusations as if an accusation is found to be false the accuser can always be traced.
- *Non-frameable official e-signatures* A group of workers in an office can all be given the authority to sign contracts and the other party to the contract knows that they have received a valid signature. The workers in the office know that they must take proper care when signing contracts as, if necessary, the signature of the contract can be traced back to them. Obviously, this could be achieved using a group signature, but knowing that the signed document came from a known set of individuals can make the recipient feel more trusting and help build the business relationship.

Since Shor [9] published efficient quantum algorithms for the solution of integer factorization and discrete logarithm problems, number-theoretic cryptography is under threat. Once a large-scale quantum computer becomes a reality then traditional cryptography schemes based on these two problems will become unsafe. The first deniable ring signature, introduced by Komano et al. [8], is based on the decisional Diffie–Hellman (DDH) problem and is not therefore quantum-resistant.

Fortunately, so far, there are a number of other cryptographic techniques that have not been successfully attacked by quantum algorithms. We focus on lattice-based schemes, as there are no known quantum algorithms for the solution of lattice-based problems and it is assumed that these will still be secure in the post-quantum age.

The first deniable ring signature proposed by Komano et al. [8] is an interactive one. For the applications above, we aim to develop a non-interactive lattice-based deniable ring signature (*NDRS*) scheme. Non-interactivity means that there is no confirmation and disavowal protocol between a ring member and the verifier. Instead the ring member provides evidence (a signature) that the verifier uses to check confirmation, or disavowal.

1.1 Contributions

We propose the first lattice-based Non-interactive Deniable Ring Signature (*NDRS*) scheme based on the ring signature scheme by Aguilar-Melchor et al. [10]. The security of our proposed scheme is based upon the SVP_γ hard lattice problem, so the scheme is believed to be quantum-resistant.

In their paper, Komano et al. outlined the security properties for a deniable ring signature, these are: anonymity, traceability and non-frameability. Our scheme also has these properties. As our scheme is non-interactive, we modify the security model in [8] to allow for this and provide a security proof.

1.2 Related works

The first ring signature scheme was introduced by Rivest et al. [11]. Komano et al. [8] introduced the concept of deniable ring signatures in 2006, basing their work on the ring signature schemes of Bellare et al. [12].

The seminal work of Ajtai [13] provided the first worst-case to average-case reduction for lattice problems, since then researchers have begun to construct many other cryptographic protocols with security based on worst-case lattice problems.

In 2010, Brakerski and Kalai [14] defined a new notion, a ring trapdoor function based on the Small Integer Solution (SIS) problem, and a framework for developing ring signatures from ring trapdoor functions in the standard model using the hash-and-sign approach [15,16]. Almost at the same time, Wang et al. [17] proposed a lattice-based ring signature scheme based on the bonsai tree signature scheme in the standard model.

In 2011, Wang and Sun [18] gave two ring signature schemes, one under the random oracle model and the other in the standard model.

In 2013, Aguilar-Melchor et al. [10] developed a ring signature scheme based on the work on digital signatures

published by Lyubachevsky [19]. This was based upon the SVP_γ hard lattice problem. Their ring signature scheme provides short secret and public keys as well as anonymity under full key exposure and unforgeability against arbitrary chosen sub-ring attacks or insider corruption in log-size rings.

In 2016, Libert et al. [20] proposed the first lattice-based ring signature with logarithmic size in the cardinality of the ring.

As far as we are aware, no one has published any work on lattice-based deniable ring signatures. While our scheme is based on the scheme of Aguilar-Melchor et al. [10], adding deniability to Libert et al's scheme [20] is the future work.

In 2017, Yoo et al. [21] proposed a general-purpose signature scheme based on supersingular elliptic curve isogenies by applying a transformation technique to the zero-knowledge proof of identity in [22]. Their scheme has small key sizes and workable performance, but large signature sizes. How to use their signature scheme as a building block to create a ring signature scheme and further to develop a deniable ring signature scheme based on supersingular elliptic curve isogenies rather than lattices is an interesting challenge. We will also consider this in our future work.

2 Preliminaries

2.1 Notation

Throughout this paper, we make use of the following notation.

$[d]$	for a positive integer d, $[d]$ is the set $\{1, \ldots, d\}$.
$\lvert S\rvert$	for a given set S, $\lvert S\rvert$ is the number of the elements in S.
$x \xleftarrow{\$} S$	x is a uniformly random sample drawn from S.
$x \leftarrow y$	Assign y to x.
k	the system security parameter; other parameters are implicitly determined by k.
n	a power of two greater than k.
c	a positive integer; the upper bound for the ring size is k^c.
p	a prime of order $\Theta(n^{4+c})$ such that $p \equiv 3 \bmod 8$.
$\mathbb{Z}_p$	the quotient ring $\mathbb{Z}/p\mathbb{Z}$.
$\mathbb{D}$	the quotient polynomial ring $\mathbb{Z}_p[x]/\langle x^n+1\rangle$. As n is a power of two, x^n+1 is irreducible. Elements in $\mathbb{D}$ are represented by polynomials of degree $n-1$ with integer coefficients in $\{-\frac{p-1}{2}, \ldots, \frac{p-1}{2}\}$.
$\mathbb{L}$	the ideal lattice ($\mathbb{L} \subseteq \mathbb{Z}_p^n$) obtained by mapping from ideals in the ring $\mathbb{D}$. This mapping is straightforward as the coefficients of the polynomials in $\mathbb{D}$ map directly to elements of vectors in $\mathbb{Z}_p^n$.
$(a, \ldots)$	the roman letters $(a, b, \ldots)$ represent polynomials.
$\lVert a\rVert_\infty$	the infinity norm of polynomial a, $\lVert a\rVert_\infty = \max_i(\lvert a^i\rvert)$ where a^i are the coefficients of the polynomial.
$(\hat{a}, \ldots)$	roman letters with a hat $(\hat{a}, \hat{b}, \ldots)$ represent vectors of polynomials, so $\hat{a} = (a_1, \ldots, a_m)$ is a vector of polynomials where m is some positive integer and $a_1, \ldots, a_m$ are polynomials.
$\lVert\hat{a}\rVert_\infty$	the infinity norm of the vector of polynomials $\hat{a}$, $\lVert\hat{a}\rVert_\infty = \max_i \lVert a_i\rVert_\infty$.
$negl(k)$	a function $f(k)$ which meets $f(k) < k^{-\tau}$ for all positive τ and sufficiently large k. Such a function is said to be *negligible*. A probability is *overwhelming* if it is $1 - negl(k)$.

Other notation will be introduced as necessary.

2.2 Collision-resistant hash functions

Lyubashevsky and Micciancio [23] introduced a family of collision-resistant hash functions based on the worst-case hardness of standard lattice problems over ideal lattices. We recall the definitions from their work.

Definition 1 ([23]) For any integer m and $D_h \subseteq \mathbb{D}$, let

$$\mathbf{H}(\mathbb{D}, D_h, m) = \{h_{\hat{a}} : \hat{a} \in \mathbb{D}^m\}$$

be the family of functions such that for any $\hat{z} \in D_h^m$,

$$h_{\hat{a}}(\hat{z}) = \hat{a} \cdot \hat{z} = \sum_{i \in [m]} a_i z_i,$$

where $\hat{a} = (a_1, \ldots, a_m)$ and $\hat{z} = (z_1, \ldots, z_m)$ and all the operations $a_i z_i$ are performed in the ring $\mathbb{D}$.

Note that for any $\hat{y}, \hat{z} \in D_h^m$ and $c \in D_h$, hash functions in $\mathbf{H}(\mathbb{D}, D_h, m)$ have the following two properties:

$$h_{\hat{a}}(\hat{y} + \hat{z}) = h_{\hat{a}}(\hat{y}) + h_{\hat{a}}(\hat{z}),$$
$$h_{\hat{a}}(\hat{y}c) = h_{\hat{a}}(\hat{y})c$$

This function family is collision resistant when the input domain is restricted to a suitably chosen subset of $\mathbb{D}^m$. Theo-

rem 1 below allows us to put limits on this subset. First recall some definitions from Lyubashevsky and Micciancio [23].

Definition 2 [short vector problem (SVP)] For $\gamma \geq 1$, a monic polynomial f and a lattice $\mathbb{L}$ corresponding to an ideal in the ring $\mathbb{Z}[x]/\langle f \rangle$, the $SVP_\gamma(\mathbb{L})$ problem seeks to find an element $g \in \mathbb{L}$ such that $\|g\|_\infty \leq \gamma \lambda_1^\infty$, where λ_1 is the size of the shortest nonzero vector on $\mathbb{L}$.

Definition 3 (collision problem) Given an element $h_{\hat{a}} \in \mathbf{H}(\mathbb{D}, D_h, m)$, the collision problem $Col(h_{\hat{a}}, D_h)$ (where $D_h \subset \mathbb{D}$) asks to find distinct elements $\hat{z}_1$ and $\hat{z}_2$ belonging to D_h such that $h_{\hat{a}}(\hat{z}_1) = h_{\hat{a}}(\hat{z}_2)$.

It was shown in [23] that, when D_h is a set of small norm polynomials, solving $Col(h_{\hat{a}}, D_h)$ is as hard as solving $SVP_\gamma(\mathbb{L})$ in the worst case over lattices that corresponding to ideals in $\mathbb{D}$.

Theorem 1 (hardness of collision-resistant hash function [23]) *Let $\mathbb{D}$ be the ring $\mathbb{Z}_p/\langle x^n + 1\rangle$ for n a power of two. Define the set $D_h = \{y \in \mathbb{D} : \|y\|_\infty \leq d\}$ for some integer d. Let $\mathbf{H}(\mathbb{D}, D_h, m)$ be a hash function family as in Definition 1 such that $m > \frac{\log p}{\log 2d}$ and $p \geq 4dmn^{1.5} \log n$. If there is a polynomial-time algorithm that solves $Col(h_{\hat{a}}, D_h)$ for random $h_{\hat{a}} \in \mathbf{H}(\mathbb{D}, D_h, m)$ with some non-negligible probability, then there is a polynomial-time algorithm that can solve $SVP_\gamma(\mathbb{L})$ for every lattice corresponding to an ideal in $\mathbb{D}$, where $\gamma = 16dmn \log^2 n$.*

Following Lyubachevsky's work [19], we set $d = mn^{1.5} \log n + \sqrt{n} \log n$. This ensures that the conditions required by the above theorem are met and that finding collision for $H \in \mathbf{H}(\mathbb{D}, D_h, m)$ implies an algorithm for breaking SVP in the worst case over ideal lattices for polynomial gaps.

2.3 Statistical distance

Statistical distance is a measure of the difference between two probability distributions. In order to be employed in the anonymity of our scheme, we recall it in this section.

Definition 4 (statistical distance) X and X' are two random variables over a countable set S. The statistical distance between X and X' is defined by

$$\Delta\left(X, X'\right) = \frac{1}{2} \sum_{x \in S} \left| \Pr\left[X = x\right] - \Pr\left[X' = x\right] \right|.$$

The following proposition shows that the statistical distance can not be increased by a randomized algorithm.

Proposition 1 (Proposition 8.10 of [24]) *Assume X and X' are two random variables over set S,*

$$\Delta\left(f(X), f\left(X'\right)\right) \leq \Delta\left(X, X'\right)$$

holds for any function f with domain S.

That is to say, if the statistical distance of two families of random variables (X_k) and (X'_k) is negligible, an adversary given a sample has negligible advantage over a wild guess in distinguishing the distributions of (X_k) and (X'_k). However, the statistical distance may increase when we consider multiple variables. From Definition 4, if X, Y come from a distribution ϕ and X', Y' a distribution ϕ', it can be verified that

$$2\Delta\left(X, X'\right) \geq \Delta\left((X, Y), \left(X', Y'\right)\right) \geq \Delta\left(X, X'\right).$$

Therefore, an adversary given more samples from the same distribution may have an increased advantage in distinguishing the distributions. If the families of random variables have an upper-bound $\epsilon(k)$ on the statistical distance, the adversary given s samples of the same distribution has an advantage bounded by $s * \epsilon(k)$.

3 Syntax and security properties of an *NDRS* scheme

In this section, we introduce the syntax and security properties of a Non-interactive Deniable Ring Signature (*NDRS*) scheme. The deniable ring signature scheme introduced by Komano et al. [8] was interactive, i.e., given a ring signature, it needs an interactive confirmation and disavowal protocol between a prover (that is a ring member) and a verifier. In this work, we describe a non-interactive deniable ring signature scheme, in which confirmation and disavowal is achieved by using another digital signature (which we call "evidence") provided by the ring members associated with the given ring signature.

3.1 Syntax

A non-interactive deniable ring signature scheme is implemented using a set of six algorithms, $NDRS = \{$ParamGen, KeyGen, Sign, Verify, EvidenceGen, EvidenceCheck$\}$.

- Setup(1^k). This setup algorithm generates the system parameters, and it takes as input a positive integer k that is a security parameter, and outputs a set of public system parameters denoted by $\mathbb{P}$.
- KeyGen$(\mathbb{P})$. This key generation algorithm takes as input the system parameters $\mathbb{P}$, and outputs a public/secret key pair for each possible signer. If the index of the signer is i, the key pair is denoted by (pk_i, sk_i).
- Sign$(\mathbb{P}, R, sk_j, \mu)$. This signing algorithm takes as input the system parameters $\mathbb{P}$, a set of l public keys R belonging to l ring members (for simplicity, let $R = (pk_1,$

$\ldots, pk_l)$), a secret key of a real signer ($j \in [l]$) sk_j and a message to be signed μ, and outputs a ring signature σ.
- Verify($\mathbb{P}, R, \mu, \sigma$). This verification algorithm takes as input the system parameters $\mathbb{P}$, the set of ring member public keys R, the message to be signed μ and the ring signature σ, and outputs "accept" or "reject".
- EvidenceGen($\mathbb{P}, R, sk_i, \mu, \sigma$). This evidence generation algorithm takes as input the system parameters $\mathbb{P}$, the set of ℓ ring member public keys R, a secret key of the evidence generator ($i \in [\ell]$) sk_i, the message to be signed μ and the ring signature σ, and outputs a piece of evidence ξ_i.
- EvidenceCheck($\mathbb{P}, R, i, \xi_i, \mu, \sigma$). This evidence checking algorithm takes as input the system parameters $\mathbb{P}$, the set of ring member public keys R, the index of the ring member, the evidence ξ_i, the message that was signed μ and the signature σ, and outputs "confirmation", "disavowal" or "failed".

3.2 Security properties

The deniable ring signature scheme described by Komano et al. [8] had the following properties:

- *Correctness* The scheme is correct provided that any ring signature generated by the signing algorithm properly is accepted by the verification algorithm and the signer of the ring signature is identified by the confirmation/disavowal protocol.
- *Anonymity* This property is preserved provided that a ring signature hides the identity of the real signer within all the ring members. This condition will, of course, be negated if the ring members are required to confirm or disavow their part in any signature.
- *Traceability* This property is preserved provided that any adversary cannot produce a ring signature that can pass the verification algorithm, but with this signature, no entity is detected as the real signer by the confirmation/disavowal protocol.
- *Non-frameability* This property is preserved provided that any adversary cannot produce a ring signature that can pass the verification algorithm, but with this signature, an entity, whose signing key is not known by the adversary, is detected as the real signer by the confirmation/disavowal protocol.

We will show that the non-interactive deniable ring signature scheme, denoted by *NDRS* and proposed in this paper, also holds these properties. The major difference is that the confirmation/disavowal protocol is replaced with an evidence generation and verification process. To define the security properties for such a scheme, we first describe the oracles that are used to address the capabilities of an adversary in a game-based security model.

Suppose that each potential ring member has a public key infrastructure (PKI) supported public/private key pair. Let List be a list issued by the PKI, and MList be a list of malicious signers who are corrupted or registered by an adversary. A signer included in List but not in MList is expected to be an honest signer. Let GSet be a list of message-signature pairs generated by a challenge oracle query $\mathsf{Ch_b}(\cdot)$. The adversary is allowed to make queries to the following oracles:

- Add(i) : The adding user oracle with the input i is invoked to add an honest signer with identity i to List. If a signer with identity i already exists, then the oracle returns ϵ that indicates the query is invalid. Otherwise, the oracle runs the key generation algorithm to create a public/secret key pair (pk_i, sk_i) for the signer, adds the signer along with the key to List, and then returns pk_i.
- Reg(i, pk_i) : Using the signer register oracle with the input of the identity i and the corresponding public key pk_i, an adversary can register a new signer i with the public key pk_i in List. The oracle also adds the signer to MList.
- Crpt(i) : The corrupting oracle with the input i is utilized to corrupt the signer whose identity is i. An adversary can draw the secret key sk_i of the signer from the oracle. If the signer i does not exist yet, the oracle can first call the adding oracle internally and then respond to the corrupting oracle. The oracle also adds the signer to MList.
- DRSig($i_k; M, i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_l$) : The deniable ring signing oracle is given the identity of a real signer with index i_k, a message M, and identities of a set of entities $i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_l$, who with i_k form a ring, and outputs a deniable ring signature σ associated with the ring.
- $\mathsf{Ch_b}(i_0, i_1, M)$: The challenge oracle is utilized in the definition of the anonymity. Given two indexes (i_0, i_1), a message M and a challenge bit $b \in \{0, 1\}$, the oracle returns a target non-interactive deniable ring signature Sign($\mathbb{P}, \{pk_{i_0}, pk_{i_1}\}, sk_{i_b}, M$) on the message M with the signer ring members of i_0 and i_1. The challenge oracle adds the target signature to GSet. Note that an adversary cannot corrupt either of the signers i_0 and i_1; moreover, the adversary cannot access the EGen query for a target signature within GSet.
- EGen(i, M, σ) : The evidence generation oracle, given the identity i and message-signature pair (M, σ), where σ is a deniable ring signature and i is one of the associated ring members, returns a piece of evidence demonstrating whether the entity i is the real signer of the signature σ or not. This oracle will reject the query if the signature being input is an output from the challenge oracle in the experiment of anonymity.

– Experiment $\mathsf{Exp}^{corr}_{NDRS,\mathbb{A}}(k)$
  $\mathsf{List} \leftarrow \emptyset$; $\mathsf{MList} \leftarrow \emptyset$; $\mathsf{GSet} \leftarrow \emptyset$;
  $(i_k; i_1, \ldots, i_l) \leftarrow$
    $\mathbb{A}(\mathsf{Add}(\cdot), \mathsf{Reg}(\cdot), \mathsf{Crpt}(\cdot), \mathsf{DRSig}(\cdot), \mathsf{EGen}(\cdot))$;
  If $i_1 \ldots i_l \notin \mathsf{List} \setminus \mathsf{MList}$ Then return 0;
  $\sigma \leftarrow \mathsf{Sign}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, sk_{i_k}, M)$,
  If $\mathsf{Verify}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, M, \sigma) = 0$ Then return 1;
  $\xi_{i_k} \leftarrow \mathsf{EvidenceGen}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, sk_{i_k}, M, \sigma)$,
  If $\mathsf{EvidenceCheck}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, i_k, \xi_{i_k}, M, \sigma) =$
    "disavowal" Then return 1;
  If $\forall sk_{i_j} \neq sk_{i_k}$
    $\xi_{i_j} \leftarrow \mathsf{EvidenceGen}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, pk_{i_j}, M, \sigma)$,
    $\mathsf{EvidenceCheck}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, i_j, \xi_{i_j}, M, \sigma) =$
      "disavowal"
  Then return 0 Else return 1

– Experiment $\mathsf{Exp}^{anon-b}_{NDRS,\mathbb{A}}(k)$
  $\mathsf{List} \leftarrow \emptyset$; $\mathsf{MList} \leftarrow \emptyset$; $\mathsf{GSet} \leftarrow \emptyset$;
  $d \leftarrow \mathbb{A}(\mathsf{Add}(\cdot), \mathsf{Reg}(\cdot), \mathsf{Crpt}(\cdot), \mathsf{Ch_b}(\cdot), \mathsf{DRSig}(\cdot), \mathsf{EGen}(\cdot))$;
  If $i \in \mathsf{GSet}$ and $i \in \mathsf{MList}$ Then return 0 Else return d

– Experiment $\mathsf{Exp}^{trace}_{NDRS,\mathbb{A}}(k)$
  $\mathsf{List} \leftarrow \emptyset$; $\mathsf{MList} \leftarrow \emptyset$; $\mathsf{GSet} \leftarrow \emptyset$;
  $(M, \sigma, \{pk_{i_1}, \ldots, pk_{i_l}\}) \leftarrow$
    $\mathbb{A}(\mathsf{Add}(\cdot), \mathsf{Reg}(\cdot), \mathsf{Crpt}(\cdot), \mathsf{DRSig}(\cdot), \mathsf{EGen}(\cdot))$;
  If $\mathsf{Verify}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, M, \sigma) = 0$ Then return 0;
  $\xi_{i_k} \leftarrow \mathsf{EvidenceGen}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, sk_{i_k}, M, \sigma)$;
  If $\forall i_j \in i_1, \ldots, i_l$
    $\mathsf{EvidenceCheck}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, i_j, \xi_{i_k}, M, \sigma) =$
      "disavowal"
  Then return 1 Else return 0

– Experiment $\mathsf{Exp}^{nf}_{NDRS,\mathbb{A}}(k)$
  $\mathsf{List} \leftarrow \emptyset$; $\mathsf{MList} \leftarrow \emptyset$; $\mathsf{GSet} \leftarrow \emptyset$;
  $(M, \sigma, \{pk_{i_1}, \ldots, pk_{i_l}\}) \leftarrow$
    $\mathbb{A}(\mathsf{Add}(\cdot), \mathsf{Reg}(\cdot), \mathsf{Crpt}(\cdot), \mathsf{DRSig}(\cdot), \mathsf{EGen}(\cdot))$
  If $\mathsf{Verify}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, M, \sigma) = 0$ Then return 0
  If the following two conditions are satisfied
  Then return 1 Else return 0
    - For some $t \in [l]$,
      $\xi_{i_t} = \mathsf{EvidenceGen}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, sk_{i_t}, M, \sigma)$,
      $\mathsf{EvidenceCheck}(\mathbb{P}, \{pk_{i_1}, \ldots, pk_{i_l}\}, i_t, \xi_{i_t}, M, \sigma) =$
        "confirmation"
    - $\mathbb{A}$ did not query $\mathsf{DRSig}(i_t; M, i_1, \ldots, i_{t-1}, i_{t+1}, \ldots, i_l)$, $\mathsf{Reg}(i_t)$ or $\mathsf{Crpt}(i_t)$

Fig. 1 Experiments of correctness, anonymity, traceability and non-frameability

– $\mathsf{Hash}(m)$: If security of an *NDRS* scheme is based on a random oracle model, this oracle, given a input data string with an arbitrary length, outputs a random number with a fixed length.

Now we are ready to define the security properties, each of which is formalized using an experiment as shown in Fig. 1. The access to the $\mathsf{Hash}(\cdot)$ oracle is the scheme specific, so we do not list it in this figure.

3.2.1 Correctness

An *NDRS* scheme is correct if:

– the signature σ generated by the Sign algorithm properly is accepted by the Verify algorithm;
– the real signer of the signature σ is identified by the output of the EvidenceGen algorithm;
– the non-real signer of the signature σ is cleared by the output of the EvidenceGen algorithm.

For an *NDRS* scheme, an adversary $\mathbb{A}$, and a security parameter k, the correctness is formalized by an experiment $\mathsf{Exp}^{corr}_{NDRS,\mathbb{A}}(k)$ in Fig. 1. The advantage of $\mathbb{A}$ is defined by

$$\mathsf{Adv}^{corr}_{NDRS,\mathbb{A}}(k) = \Pr\left[\mathsf{Exp}^{corr}_{NDRS,\mathbb{A}}(k) = 1\right].$$

The *NDRS* is correct if $\mathsf{Adv}^{corr}_{NDRS,\mathbb{A}}(k)$ is negligible for any probabilistic polynomial-time adversary $\mathbb{A}$ and security parameter k.

3.2.2 Anonymity

A deniable ring signature does not reveal who from the set of the ring members is the real signer. For an *NDRS* scheme, a security parameter k, a positive integer l indicating the number of potential ring members, and a PPT adversary $\mathbb{A}$, the property of anonymity is formalized using the experiment $\mathsf{Exp}^{anon-b}_{NDRS,\mathbb{A}}(k)$ as described in Fig. 1. The advantage $\mathsf{Adv}^{anon}_{NDRS,\mathbb{A}}(k)$ is defined as

$$\begin{aligned}\mathsf{Adv}^{anon}_{NDRS,\mathbb{A}}(k) &= \left|\Pr\left[\mathsf{Exp}^{anon-1}_{NDRS,\mathbb{A}}(k) = 1\right]\right| \\ &\quad - \left|\Pr\left[\mathsf{Exp}^{anon-0}_{NDRS,\mathbb{A}}(k) = 1\right]\right| \\ &= \left|2\Pr\left[\mathsf{Exp}^{anon-b}_{NDRS,\mathbb{A}}(k) = b\right] - 1\right|.\end{aligned}$$

More specifically, a non-interactive deniable ring signature scheme is said to be anonymous in $(\tau, q_{Ch}, q_H, q_S, q_E, q_C, \epsilon)$ if the advantage is less than ϵ for any $\mathbb{A}$, with the time bound τ, and querying the challenge oracle, hash oracle, signing oracle, evidence generation oracle up to q_{Ch}, q_H, q_S, q_E times, respectively. Note that if the system includes l signers, $\mathbb{A}$ can query the signer register oracle and corrupt oracle at most $l-2$ times in total; this value is the upper bound of the sum of the times of these two queries q_C.

Note that in the literature, there are many different definitions of anonymity, for example k-anonymity [25] and l-diversity [26]. Both of these techniques are designed to measure the data privacy level in databases. A database table T satisfies k-anonymity if for every item $t \in T$ there exist at least $k-1$ other items in T which share the same set of attributes with t. A ring signature with ℓ ring members achieves ℓ-anonymity. For a deniable ring signature with ℓ ring members, before any member provides evidence by following the EvidenceGen algorithm, the signature achieves ℓ-anonymity; after $n < (\ell - 1)$ non-real signers provide

their evidence, the signature achieves $(\ell - n)$-anonymity; after $\ell - 1$ non-real signers or the real signer provide their evidence, the signature is reduced to an ordinary digital signature without anonymity. As discussed in [26], k-anonymity is a weak measurement of data privacy, and l-diversity is suggested to increase the privacy level by reducing the granularity of data representation. If we replace ring signatures with group signatures [4,5,12], we can reduce granularity, since a group signature does not reveal the details of the group. However, this change is not cost free, as we lose the flexibility provided by ring signatures.

3.2.3 Traceability

For a non-interactive deniable ring signature scheme *NDRS*, any adversary $\mathbb{A}$ and security parameter k, the property of traceability is formalized using the experiment $\mathsf{Exp}^{trace}_{NDRS,\mathbb{A}}(k)$ as shown in Fig. 1. The advantage of the adversary is defined as:

$$\mathsf{Adv}^{trace}_{NDRS,\mathbb{A}}(k) = \Pr\left[\mathsf{Exp}^{trace}_{NDRS,\mathbb{A}}(k) = 1\right].$$

More specifically, the *NDRS* scheme is said to hold traceability in $(\tau, q_H, q_S, q_E, \epsilon)$ if the advantage is less than ϵ for any adversary $\mathbb{A}$, with time bound τ, and querying the hash oracle, signing oracle, and evidence generation oracle q_H, q_S and q_E times, respectively. If the system includes l signers, $\mathbb{A}$ can query the signer register oracle and corrupt oracle at most $l - 1$ times in total.

3.2.4 Non-frameability

For an *NDRS* scheme, any adversary $\mathbb{A}$ and security parameter k, the property of non-frameability is formalized using the experiment $\mathsf{Exp}^{nf}_{NDRS,\mathbb{A}}(k)$ as shown in Fig. 1. The advantage of the adversary is defined as:

$$\mathsf{Adv}^{nf}_{NDRS,\mathbb{A}}(k) = \Pr\left[\mathsf{Exp}^{nf}_{NDRS,\mathbb{A}}(k) = 1\right].$$

More specifically, the *NDRS* scheme is said to hold non-frameability in $(\tau, q_H, q_S, q_E, \epsilon)$ if the advantage is less than ϵ for any adversary $\mathbb{A}$, with time bound τ, and querying the hash oracle, signing oracle, and evidence generation oracle q_H, q_S and q_E times, respectively. If the system includes l signers, $\mathbb{A}$ can query the signer register oracle and corrupt oracle at most $l - 1$ times in total.

4 Construction of *NDRS*

In this section, we present our construction of *NDRS* from lattices. Our scheme is an extension of the ring signature scheme of Aguilar-Melchor et al. [10] (referred to as AM in what follows) designed to achieve the property of non-interactive deniability. As it is shown in Fig. 2, our construction of *NDRS* consists of six algorithms, Setup, KeyGen, Sign, Verify, EvidenceGen and EvidenceCheck, which are now described and, where appropriate, compared with those in AM.

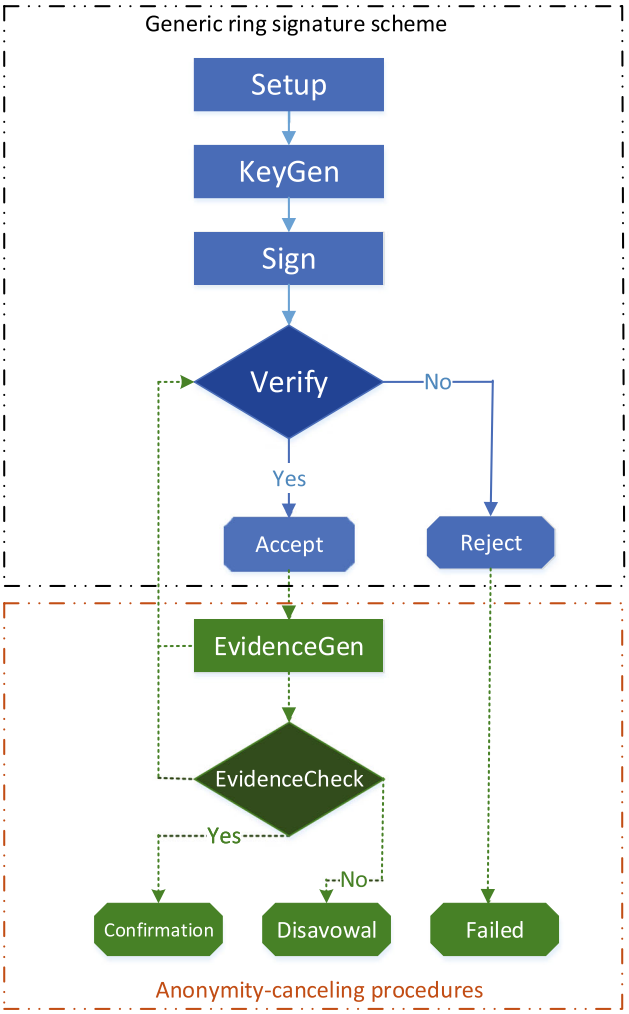

Fig. 2 Our construction of *NDRS*

4.1 Setup(1^k)

This algorithm takes as input an integer k that is a security parameter, and outputs the system parameters $\mathbb{P} = (k, n, p, m, \mathbb{D}, D_h, D_y, D_z, D_s, S, \mathbb{H}, H_1, H_2, H_3)$, as listed in Table 1. Apart from the three hash functions, all of the parameters are taken from AM. Attack is easier as the ring size (i.e., the number of public keys used in a ring signature) grows (also true for other schemes) and so, as in AM, we use a constant c such that acceptable ring sizes are bounded from above by k^c. c is used to increase some of the parameters

Table 1 The system parameters

k	integer security parameter
n	integer: power of 2 greater than k
p	prime: order of $\Theta(n^{4+c})$ such that $p \equiv 3 \mod 8$
m	integer: $(3+2c/3)\log n$
$\mathbb{D}$	quotient ring of polynomials: $\mathbb{Z}_p[x]/\langle x^n+1\rangle$
D_h	$\{g \in \mathbb{D} : \|g\|_\infty \leq mn^{1.5}\log n + \sqrt{n}\log n\}$
D_y	$\{g \in \mathbb{D} : \|g\|_\infty \leq mn^{1.5}\log n\}$
D_z	$\{g \in \mathbb{D} : \|g\|_\infty \leq mn^{1.5}\log n - \sqrt{n}\log n\}$
D_s	$\{g \in \mathbb{D} : \|g\|_\infty \leq 1\}$
S	an element of $\mathbb{D}$: $S \leftarrow \mathbb{D}$ and $S \neq 0$
$\mathbb{H}$	a family of hash functions: $D_h^m \rightarrow \mathbb{D}$
H_1	a hash function: $\{0,1\}^* \rightarrow D_s$
H_2	a hash function: $\{0,1\}^* \rightarrow D_s$
H_3	a hash function: $\{0,1\}^* \rightarrow D_s$

compared to those given in [27] because the use of the ring makes the system more vulnerable to attack. AM states that a c value of 1 or 2 is sufficient to cover any reasonable use of these signatures.

4.2 KeyGen($\mathbb{P}$)

Let N be the total number of possible signers. For simplicity, let the index of each signer be $i \in \{1, 2, \ldots, N\}$ and let (pk_i, sk_i) be i's public and secret key pair. Let K_p be the set of public keys for all of the possible signers, $K_p = (pk_1, \ldots, pk_N)$. This set is accessible to any signer.

Each possible signer uses this algorithm to generate their public and secret keys and adds their public key to K_p. To generate the public and secret keys for signer i this algorithm takes as input the system parameters $\mathbb{P}$, and performs the following steps:

1. Set $\hat{s_i} = (s_1, s_2, \ldots, s_m) \xleftarrow{\$} D_s^m$. For $t \in [m]$, if none of the values s_t is invertible, reset $\hat{s_i}$; otherwise, let $t_0 \in \{1, \ldots, m\}$ such that s_{t_0} is invertible.
2. Set $\hat{a_i} = (a_1, \ldots, a_m)$, such that $(a_1, a_2, \ldots, a_{t_0-1}, a_{t_0+1}, \ldots, a_m) \xleftarrow{\$} \mathbb{D}^{m-1}$ and $a_{t_0} = s_{t_0}^{-1}(S - \sum_{t \neq t_0} a_t s_t) \in \mathbb{D}$.
3. Let $h_{\hat{a_i}}(.) \in \mathbb{H}$ defined by $\hat{a_i}$; e.g., $h_{\hat{a_i}}(\hat{s_i}) = \sum_t a_t \cdot s_t = S$, where $a_t \cdot s_t$ is polynomial multiplication in $\mathbb{D}$.
4. Output $(pk_i, sk_i) = (\hat{a_i}, \hat{s_i})$.

This algorithm is the same as that in AM.

4.3 Sign($\mathbb{P}$, R, sk_j, μ)

Assume that a real signer $j \in [N]$ chooses $l-1$ other possible signers to form a signer ring and creates a ring signature; we refer this ring as $U \subset [N]$ with l members, and for simplicity, we denote each ring member by $i \in [l]$ taken in the order of index value. Let R be the public keys of the ring members, $R = (\hat{a_1}, \ldots, \hat{a_l})$ and $\tilde{R} = \hat{a_1}||\cdots||\hat{a_l}$, the concatenation of these public keys. The algorithm Sign takes as input the system parameters $\mathbb{P}$, the set of public keys R, the secret key of the real signer, $sk_j = \hat{s_j}$ for $j \in [l]$, and a message to be signed μ, and outputs a ring signature σ or an error message "failed" by performing the following steps:

1. Check if each parameter in $\mathbb{P}$ satisfies the definition described in Table 1, sk_j is in D_s^m, the size of signer ring l is bounded by k^c, and each public key $\hat{a_i}$ in R is in $\mathbb{D}^m$. Output "failed" if any the above check is not passed.
2. Set $\hat{b} \xleftarrow{\$} \mathbb{D}^m$ and let $\hat{b}$ determine a hash function $h_{\hat{b}}(\cdot) \in \mathbb{H}$.
3. Compute $\sigma_j = h_{\hat{b}}(\hat{s_j})$. If $\sigma_j = 0 \mod S$, return to step 2 to reset $\hat{b}$; otherwise compute $A = \sigma_j - H_1(j||\hat{a}_j)\cdot S \in \mathbb{D}$.
4. For j, generate $\hat{y}_j \xleftarrow{\$} D_y^m$, and then compute $\alpha_j = h_{\hat{a}_j}(\hat{y_j})$ and $\beta_j = h_{\hat{b}}(\hat{y_j})$.
5. $\forall i (i \neq j) \in [l]$, generate $\hat{z}_i \xleftarrow{\$} D_z^m$ and $v_i \xleftarrow{\$} D_s$, and then compute $\sigma_i = H_1(i||\hat{a}_i) \cdot S + A \in \mathbb{D}$, $\alpha_i = h_{\hat{a}_i}(\hat{z_i}) - S \cdot v_i \in \mathbb{D}$ and $\beta_i = h_{\hat{b}}(\hat{z_i}) - \sigma_i \cdot v_i \in \mathbb{D}$.
6. Compute $v = H_2(\sum_{i\in[l]} \alpha_i, \tilde{\beta}, A, \tilde{R}, \mu) \in D_s$, where $\tilde{\beta} = \hat{\beta_1}||\cdots||\hat{\beta_l}$ and $\tilde{R} = \hat{a_1}||\cdots||\hat{a_l}$.
7. Compute $v_j = v - \sum_{i\neq j, i\in[l]} v_i$ and $\hat{z_j} = \hat{y_j} + \hat{s_j} \cdot v_j$. If $\hat{z_j} \in D_z^m$ or $v_j \in D_s$ does not hold, then go back to re-select any $\hat{z}_{i\neq j}$ or $v_{i\neq j}$ or $\hat{y}_j$.
8. Output $\sigma = (\hat{b}, A, \tilde{z}_U, \tilde{v}_U)$ as the ring signature on μ under R, where $\tilde{z}_U = (\hat{z_1}, \ldots, \hat{z_l})$ and $\tilde{v}_U = (v_1, \ldots, v_l)$.

Figure 3 shows a comparison between this algorithm and the signing algorithm from AM.

4.4 Verify($\mathbb{P}$, R, μ, σ)

Take as input the system parameters $\mathbb{P}$, the set of public keys R whose ring member indexes in U, a message μ and a ring signature σ, the verifier operates as follows to check the signature.

1. Parse σ as $\hat{b}, A, \tilde{z}_U = (\hat{z_1}, \ldots, \hat{z_l})$ and $\tilde{v}_U = (v_1, \ldots, v_l)$. Check that,
   - all of the inputs are in their correct places.
   - $A \neq 0 \mod S$.
   - for $\forall i$, check $\hat{z}_i \in D_z^m$.

   If any of the above checks do not pass, reject the signature.
2. For $\forall i \in [l]$, compute $\sigma_i' = H_1(i||\hat{a}_i) \cdot S + A \in \mathbb{D}$, $\alpha_i' = h_{\hat{a}_i}(\hat{z_i}) - S \cdot v_i \in \mathbb{D}$ and $\beta_i' = h_{\hat{b}}(\hat{z_i}) - \sigma_i' \cdot v_i \in \mathbb{D}$.

$\mathsf{Sign}_{\mathrm{AM}}$	$\mathsf{Sign}_{\mathrm{Ours}}$
	$\hat{b} \xleftarrow{\$} \mathbb{D}^m$, s.t. $\sigma_j = h_{\hat{b}}(\hat{s_j}) \neq 0 \mod S$
	$A = \sigma_j - H_1(j\|\|\hat{a}_j) \cdot S$
$\hat{y_j} \xleftarrow{\$} D_y^m, \alpha_j = h_{\hat{a}_j}(\hat{y_j})$	$\hat{y_j} \xleftarrow{\$} D_y^m, \alpha_j = h_{\hat{a}_j}(\hat{y_j})$
	$\beta_j = h_{\hat{b}}(\hat{y_j})$
$\forall i(\neq j) \in [l]; \hat{z_i} \xleftarrow{\$} D_z^m$	$\forall i(\neq j) \in [l], \hat{z_i} \xleftarrow{\$} D_z^m$, $v_i \xleftarrow{\$} D_s$
	$\sigma_i = H_1(i\|\|\hat{a}_i) \cdot S + A$
$\alpha_i = h_{\hat{a}_i}(\hat{z_i})$	$\alpha_i = h_{\hat{a}_i}(\hat{z_i}) - S \cdot v_i$
	$\beta_i = h_{\hat{b}}(\hat{z_i}) - \sigma_i \cdot v_i \in \mathbb{D}$
$v = H_2(\sum_{i\in[l]} \alpha_i, \tilde{R}, \mu)$	$v = H_2(\sum_{i\in[l]} \alpha_i, \tilde{\beta}, A, \tilde{R}, \mu)$, where $\tilde{\beta} = \hat{\beta}_1\|\|...\|\|\hat{\beta}_l$
	$v_j = v - \sum_{i\neq j, i\in[l]} v_i$,
$\hat{z_j} = \hat{y_j} + \hat{s_j} \cdot v$	$\hat{z_j} = \hat{y_j} + \hat{s_j} \cdot v_j$
$\sigma = (\tilde{z}_U, v)$, for $\tilde{z}_U = (\hat{z}_1, \cdots, \hat{z}_l)$	$\sigma = (\hat{b}, A, \tilde{z}_U, \tilde{v}_U)$, for $\tilde{z}_U = (\hat{z}_1, \cdots, \hat{z}_l)$, $\tilde{v}_U = (v_1, \cdots, v_l)$

Fig. 3 The signing algorithm from AM compared to ours

$\mathsf{Verify}_{\mathrm{AM}}$	$\mathsf{Verify}_{\mathrm{Ours}}$
	Check $A \neq 0 \mod S$
$\forall i \in [l]$, $\alpha'_i = h_{\hat{a}_i}(\hat{z_i})$	$\forall i \in [l]$, $\alpha'_i = h_{\hat{a}_i}(\hat{z_i}) - S \cdot v_i$
	$\sigma'_i = H_1(i\|\|\hat{a}_i) \cdot S + A$
	$\beta'_i = h_{\hat{b}}(\hat{z_i}) - \sigma'_i \cdot v_i$
$t' = \sum_{i\in[l]} \alpha'_i - S \cdot v, \tilde{R}, \mu$	$t' = \sum_{i\in[l]} \alpha'_i, \tilde{\beta}', A, \tilde{R}, \mu$, where $\tilde{\beta}' = \hat{\beta}'_1\|\|...\|\|\hat{\beta}'_l$
$v' = H_2(t')$	$v' = H_2(t')$
If $v' = v$, accept, else reject	If $v' = \sum_{i\in[l]} v_i$, accept, else reject

Fig. 4 The verification algorithm from AM compared to ours

3. Compute $v' = H_2(\sum_{i\in[i]} \alpha'_i, \tilde{\beta}', A, \tilde{R}, \mu) \in D_s$, where $\tilde{\beta}' = \hat{\beta}'_1\|\|\cdots\|\|\hat{\beta}'_l$ and $\tilde{R} = \hat{a_1}\|\|\cdots\|\|\hat{a_l}$.
4. If $v' = \sum_{i\in[l]} v_i$, accept the signature, otherwise reject it.

Figure 4 shows a comparison between this algorithm and the corresponding verification algorithm from AM.

4.5 EvidenceGen($\mathbb{P}$, R, sk_i, μ, σ)

Assume that (μ, σ) is a valid message-signature pair under the public keys of the ring members R. Any involved signer $i \in U$ can generate a piece of evidence, ξ_i, which shows whether i is a real signer or not. This algorithm takes as input the system parameters $\mathbb{P}$, the set of ring member's public keys R, i's secret key sk_i, and a message and signature pair (μ, σ), outputs the evidence by performing the following steps:

1. run $\mathsf{Verify}(\mathbb{P}, R, \mu, \sigma)$: if the result is "reject", abort and output an error message "failed"; otherwise, carry on.
2. compute $\sigma_i = h_{\hat{b}}(\hat{s_i})$.
3. set $\hat{y_i} \xleftarrow{\$} D_y^m$, and compute $\alpha_i = h_{\hat{a}_i}(\hat{y_i})$ and $\beta_i = h_{\hat{b}}(\hat{y_i})$.
4. compute $e_i = H_3(\alpha_i, \beta_i, A, \tilde{R}, \mu) \in D_s$ and $\hat{z_i} = \hat{y_i} + \hat{s_i} \cdot e_i$.
5. output $\xi_i = (\sigma_i, \alpha_i, \beta_i, \hat{z}_i, e_i)$.

4.6 EvidenceCheck($\mathbb{P}$, R, i, ξ_i, μ, σ)

This algorithm takes as input the system parameters $\mathbb{P}$, the set of ring member's public keys R, an index i, an evidence ξ_i, a message μ and a ring signature σ, and performs the following steps:

1. run $\mathsf{Verify}(\mathbb{P}, R, \mu, \sigma)$; if the result is "reject", abort and output an error message "failed"; otherwise, carry on.
2. parse ξ_i as $(\sigma_i, \alpha_i, \beta_i, \hat{z}_i, e_i)$.
3. compute $\alpha'_i = h_{\hat{a}_i}(\hat{z_i}) - S \cdot e_i \in \mathbb{D}$, $\beta'_i = h_{\hat{b}}(\hat{z_i}) - \sigma_i \cdot e_i \in \mathbb{D}$, and $e'_i = H_3(\alpha'_i, \beta'_i, A, \tilde{R}, \mu) \in D_s$.
4. check whether $e_i = e'_i$. If not, abort and return "failed".
5. check whether $\sigma_i = H_1(i\|\|\hat{a}_i) \cdot S + A$. If it holds, output "confirmation" indicating that i associated with $pk_i = \hat{a}_i$ is the real signer for σ, otherwise output "disavowal" indicating that i is not the real signer for σ.

Note that given a set of evidence $(\forall i \in U, \xi_i)$, a verifier can find who is the real signer and who is not. The verifier initiates two empty lists C and D, and repeats the above $\mathsf{EvidenceCheck}$ algorithm for each i. When the algorithm outputs "confirmation" the verifier adds i into List C, and when the algorithm outputs "disavowal" adds i into List D. In the end, the verifier outputs the final value of C and D. We will prove later that $|C| \leq 1$ always holds.

Remark 1 Just as in AM, the signature length in the *NDRS* scheme is linear with the number of ring members used. To maintain anonymity, the signer should choose as many ring members as possible (subject to the upper bound k^c). There is clearly a trade-off between anonymity, and both the computation cost and the signature length.

Remark 2 Using the same $\hat{s}_i$ for a number of calculations of $\sigma_i = h_{\hat{b}}(s_i)$ with different $\hat{b}$ values may leak information about $\hat{s}_i$. How often the same $\hat{s}_i$ can be used and what constraints need to be put on the system parameters to ensure that this does not happen is a topic for further work.

5 Security analysis

The required properties of the proposed scheme are analyzed in this section. As described in Sect. 3.2, these properties are correctness, anonymity, traceability and non-frameability. We first justify why the scheme is correct and then provide proofs of the other properties (Theorems 2, 3 and 4).

By following the description of the proposed scheme, it is easy to see that the scheme is correct. An honestly created *NDRS* signature under properly created keys will always pass the Verify algorithm. With this signature and following the EvidenceGen algorithm, the real signer is able to create a piece of evidence, which leads the EvidenceCheck algorithm to output "confirmation". In addition, a non-real signer is able to create a piece of evidence that leads the EvidenceCheck algorithm to output "disavowal".

Theorem 2 (anonymity) For $b \in \{0, 1\}$, let $X_{b,\mathbb{P},R,sk_{i_b},\mu}$ be the output of $\mathsf{Sign}(\mathbb{P}, R, sk_{i_b}, \mu)$ (responding to the $\mathsf{Ch_b}$ query) with $\mathbb{P}$, R, sk_{i_b} and μ as a set of arbitrary inputs to the algorithm. From the point of view of any probabilistic polynomial-time turning machine without the knowledge of sk_{i_0} or sk_{i_1}, the following condition holds

$$\Delta\left(X_{0,\mathbb{P},R,sk_{i_0},\mu}, X_{1,\mathbb{P},R,sk_{i_1},\mu}\right) \leq \epsilon.$$

Therefore, the proposed *NDRS* scheme is computationally anonymous in $(\tau, q_{Ch}, q_H, q_S, q_E, q_C, \epsilon)$ under the random oracle model.

Proof In the anonymity experiment in Fig. 1, the adversary $\mathbb{A}$ is a probabilistic polynomial-time turing machine, which is allowed to make queries to the following oracles:

- $\mathsf{Add}(i)$: adding a user;
- $\mathsf{Reg}(i, pk_i)$: registering a signer;
- $\mathsf{Crpt}(i)$: corrupting a signer;
- $\mathsf{DRSig}(i_k, M, i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_l)$: generating a signature;
- $\mathsf{Ch_b}(i_0, i_1, M)$: getting a challenging signature σ_b;
- $\mathsf{EGen}(i, M, \sigma)$: generating an evidence;
- $\mathsf{Hash}(m)$: getting a hash value.

This anonymity experiment starts with $\mathbf{List} \leftarrow \emptyset$, $\mathbf{MList} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$ and $\mathbf{HList} \leftarrow \emptyset$. The time bound τ, the query numbers of q_{Ch}, q_H, q_S, q_E and q_C and the ϵ value are associated with the security parameter k. Finally, $\mathbb{A}$ returns a guess of bit b (which is denoted as d in Fig. 1) according to the signature σ_b from $\mathsf{Ch_b}(i_0, i_1, M)$.

The oracles handle the queries as follows:

- $\mathsf{Add}(i)$: If $i \in \mathbf{List}$, return $\perp$; otherwise generate $h_{\hat{a}_i}$ for $\hat{s}_i \leftarrow_R D_s^m$, add $(i, \hat{s}_i, h_{\hat{a}_i})$ into **List**, and returns $h_{\hat{a}_i}$.
- $\mathsf{Reg}(i, h_{\hat{\mathbf{a}}_i})$: If $i \in \mathbf{List}$, return $\perp$; otherwise set $h_{\hat{a}_i}$ as a public key of the signer with index i, add $(i, \cdot, h_{\hat{a}_i})$ into **List** and i into **MList**.
- $\mathsf{Crpt}(i)$: If $i \notin \mathbf{List} \setminus \mathbf{MList}$, return $\perp$; otherwise add i into **MList** and return $\hat{s}_i$.
- $\mathsf{Hash}(x_\alpha, x_\beta, A, \tilde{R}, \mu; 2/3)$: If $(x_\alpha, x_\beta, A, \tilde{R}, \mu; v) \in \mathbf{HList}$, return v. If not, choose $v \leftarrow D_s$ and program $H(x_\alpha, x_\beta, A, \tilde{R}, \mu) = v$ for $H = \{H_2, H_3\}$, add $(x_\alpha, x_\beta, A, \tilde{R}, \mu; v)$ into **HList**, and return v.
- $\mathsf{DRSign}(j, \mu, U)$: $\forall i \in U$ including j, if $i \notin \mathbf{List} \setminus \mathbf{MList}$, return $\perp$. Otherwise there are two cases:
  - (i) If $sk_j \in \mathsf{List}$, following the Sign algorithm create and return a signature σ;
  - (ii) If $sk_j \notin \mathsf{List}$, choose $\hat{b}$ at random, set $\sigma_j = h_{\hat{b}}(\hat{s_j})$ at random (note that this is handled as a random oracle without the knowledge of $\hat{s_j}$). Compute $A = \sigma_j - H_1(j||\hat{a}_j) \cdot S$, choose z_i and v_i at random and set $v = \sum_i v_i$ as $H(\sum_i \alpha_i, \sum_i \beta_i, A, \tilde{R}, \mu)$ (again this is handled as a random oracle without the knowledge of α_i and β_i), then compute $\alpha_i = h_{\hat{a}_i}(\hat{z}_i) - S \cdot v_i$ and $\beta_i = h_{\hat{b}}(\hat{z}_i) - (H_1(i||\hat{a}_i) \cdot S + A) \cdot v_i$, and return the signature $\sigma = (\hat{b}, A, \tilde{z}_U, \tilde{z}_U)$.
- $\mathsf{EGen}(i, \mu, \sigma)$: If $i \notin \mathbf{List} \setminus \mathbf{MList}$, return $\perp$. Otherwise, there are two cases:
  - (i) If $sk_i \in \mathsf{List}$, following the EvidenceGen algorithm create and return the evidence σ_e;
  - (ii) If $sk_i \notin \mathsf{List}$, forge the signature σ_e by controlling the random oracle hash function H' and return the value σ_e.
- $\mathsf{Ch_b}(i_0, i_1, \mu^*, R^*)$: If i_0 or $i_1 \notin \mathbf{List} \setminus \mathbf{MList}$ return $\perp$. Otherwise, choose $\hat{b}_0, \hat{b}_1$ at random, and compute σ_{i_0} and σ_{i_0} with $\hat{s}_{i_0}$ and $\hat{s}_{i_1}$, respectively, by following the Sign algorithm. Choose $b \in \{0, 1\}$ at random and return σ_{i_b}.

Recall that our proposed scheme is an extension of the ring signature scheme by Aguilar-Melchor [10]. Our scheme can be seen as a combination of two ring signatures $\sigma = (\sigma^\alpha, \sigma^\beta)$: $\sigma^\alpha = (\tilde{z}_U, v = \sum_{i \in U} v_i)$ associated with the α_i values; $\sigma^\beta = (\hat{b}, A, \tilde{z}_U, \tilde{v}_U)$ associated with the β_i values.

It is easy to see that σ^α is the Aguilar-Melchor et al. ring signature scheme. Regarding anonymity, this signature is unconditional anonymous and holds the following theorem (Theorem 2 of [10]): For $b \in \{0, 1\}$, let $X^\alpha_{b,\mathbb{P},sk_{i_b},\mu,R}$ be the random variable describing the output of the ring signing algorithm (in the $\mathsf{Ch_b}$ query), and the following equation holds:

$$\Delta\left(X^\alpha_{0,\mathbb{P},R,sk_{i_0},\mu}, X^\alpha_{1,\mathbb{P},R,sk_{i_1},\mu}\right) = n^{-\omega(1)} \leq \epsilon_\alpha.$$

σ^{β} involves the extra values $\hat{b}$, A and v_i for $i \in [l]$ (instead of a single v value). This extra information makes our scheme is not unconditional anonymous, since, obviously, anyone with the knowledge of sk_{i_0} or sk_{i_1} can distinguish $X^{\beta}_{0,\mathbb{P},R,sk_{i_0},\mu}$ from $X^{\beta}_{1,\mathbb{P},R,sk_{i_1},\mu}$. This is designed in the purpose of supporting deniability. Now we argue that σ^{β} holds computational anonymity. This means that for a probabilistic polynomial-time turning machine adversary $\mathbb{A}$, we have

$$\Delta\left(X^{\beta}_{0,\mathbb{P},R,sk_{i_0},\mu}, X^{\beta}_{1,\mathbb{P},R,sk_{i_1},\mu}\right) \leq \epsilon_{\beta}.$$

In σ^{β}, a fresh public key σ_{i_b} is created for the real signer i_b, corresponding to the private signing key sk_{i_b} and using a new random hash function $h_{\hat{b}}()$. In order to make σ^{β} be a ring signature, a dummy public key $\sigma_{i_{\bar{b}}}$ is created for a non-real signer $i_{\bar{b}}$. The value A represents the entire set of these fresh public keys, $A = \sigma_{i_b} + H_1(i_b||\hat{a}_{i_b}) \cdot S = \sigma_{i_{\bar{b}}} + H_1(i_{\bar{b}}||\hat{a}_{i_{\bar{b}}}) \cdot S$.

Let us now discuss the statistical distance between two σ^{β} signatures under two secret keys sk_{i_0} and sk_{i_1}. For simplifying the notation, we add subscripts i_0 and i_1 for the components of signatures $\sigma^{\beta}_{i_0}$ and $\sigma^{\beta}_{i_1}$ under private keys $\hat{s}_{i_0}$ and $\hat{s}_{i_1}$. As $\hat{b}_{i_0}$ and $\hat{b}_{i_1}$ are created at random by the challenger and $\hat{a}_{i_0}$ and $\hat{a}_{i_1}$ are created consistently and randomly, we know that these values are independent of bit b, and the statistical distance between $\hat{b}_{i_0}$ and $\hat{b}_{i_1}$ is 0. Without accessing to the corresponding private keys $\hat{s}_{i_0}$ and $\hat{s}_{i_1}$, the adversary $\mathbb{A}$ can not distinguish $\sigma_{i_0} = h_{\hat{b}_{i_0}}(\hat{s}_{i_0})$ and $\sigma_{i_1} = h_{\hat{b}_{i_1}}(\hat{s}_{i_1})$ from $pk_{i_0} = \hat{a}_{i_0}$ and $pk_{i_1} = \hat{a}_{i_1}$ with probability more than ϵ, because the hash functions $h_{\hat{b}_{i_0}}(\hat{s}_{i_0})$ and $h_{\hat{b}_{i_1}}(\hat{s}_{i_1})$ are handled by the random oracle in the anonymity experiment. From proposition 8.10 in [24] (Let X, Y be two random variables over a common set A. For any function f with domain A, the statistical distance between $f(X)$ and $f(Y)$ is at most $\Delta(f(X), f(Y)) \leq \Delta(X, Y)$), we can obtain the conclusion that the statistical distance between A_{i_0} and A_{i_1} is also no more than ϵ from $A_{i_0} = h_{\hat{b}_{i_0}}(\hat{s}_{i_0}) - H_1(i_0||\hat{a}_{i_0}) \cdot S$ and $A_{i_1} = h_{\hat{b}_{i_1}}(\hat{s}_{i_1}) - H_1(i_1||\hat{a}_{i_1}) \cdot S$.

Based on the above analysis of σ^{α} and σ^{β}, we have

$$\Delta\left(X_{0,\mathbb{P},R,sk_{i_0},\mu}, X_{1,\mathbb{P},R,sk_{i_1},\mu}\right) \leq \epsilon = \epsilon_{\alpha} + \epsilon_{\beta}.$$

This completes the proof. □

Theorem 3 (traceability) *Under the assumptions that the ring signature scheme of* [10] *and the Lyubashevsky signature scheme* [19] *are unforgeable, our proposed NDRS scheme in Sect. 4 is traceable under the random oracle model.*

Proof We prove this theorem with the following steps of discussion:

- An *NDRS* signature is based on Aguilar-Melchor et al's ring signature [10] that is associated with the l-ring long-term public keys R. Via Lemma 1, we show that due to the Aguilar-Melchor et al. ring signature is unforgeable, if a given ring signature from our *NDRS* scheme can be accepted by running the Verify algorithm, it must be created by a "real-signer" under its private key corresponding to its public key that must be one from R.
- A signature from the proposed *NDRS* scheme includes another ring signature that is associated with the ephemeral l-ring public keys σ_i for $i = \{1, \ldots, l\}$. Via Lemma 2, we show that due to the Lyubashevsky signature scheme [19] is unforgeable, if a given ring signature from our *NDRS* scheme can be accepted by running the Verify algorithm, it must be created by a "real-signer" under its private key corresponding to its ephemeral public key that must be one from the l-ring public keys σ_i for $i \in \{1, \ldots, l\}$.
- A Schnorr-type of signature-based knowledge proof is used in our *NDRS* scheme as a connection between these two ring signatures. We show that due to the unforgeability of these two ring signatures and the nature of the Schnorr signature-based proof, this connection indicates that these two ring signatures must share the same real-signer, say i. This means that one of the σ_i value is a real public key corresponding to the real private key $sk_i = \hat{s}_i$. The difference is that the long-term public key pk_i is associated with the long-term base $\hat{a}_i$, (i.e., $S = h_{\hat{a}_i}(\hat{s}_i)$) and the ephemeral public key is associated with the random base $\hat{b}$ (i.e., $\sigma_i = h_{\hat{b}}(\hat{s}_i)$).
- The evidence in the proposed *NDRS* scheme is based on the Lyubashevsky lattice-based signature scheme [19], and it includes two Lyubashevsky signatures, both in the Schnorr-type of signature-based knowledge proof format. Using Lemma 3, we show that the evidence is also unforgeable assume that the Lyubashevsky signature is unforgeable. Following the same observation discussed before, these two signatures in the same evidence share the same private key.

Finally, we conclude that the ring signature and the evidence from the real signer i must share the same σ_i value. Therefore, the ring signature with the associated evidence is traceable.

Next, let us describe and prove these three lemmas.

Lemma 1 *If there is a polynomial-time algorithm that can forge a ring signature from the NDRS scheme, there is another polynomial-time algorithm that can forge a ring signature from the Aguilar-Melchor et al. ring signature scheme in* [10].

Proof Recall our ring signature scheme is an extension of the ring signature scheme in [10] by adding $\hat{b}$, σ_i, A and β_i for $i \in \{1, \ldots, l\}$. If there is an adversary $\mathbb{A}$ who is able to forge

a ring signature from our construction, there exists another adversary $\mathbb{B}$ who can play the role as $\mathbb{A}$'s challenger and use $\mathbb{A}$'s response to forge a ring signature of [10], provided that $\mathbb{B}$ is able to control the random oracle H used by $\mathbb{A}$.

The Aguilar-Melchor et al. ring signature scheme has the same public and private key setting as our *NDRS* scheme. Let $\sigma_{\mathbb{B}} = (\tilde{z}_U, e)$ be a ring signature from [10], where $e = H(\sum_{i\in[l]} h_{\hat{a}_i}(\hat{z}_i) - S \cdot e, R, \mu))$ and let $\sigma_{\mathbb{A}} = (\hat{b}, \tilde{z}_U, A, \tilde{v}_U)$ be a ring signature from our construction, with $v = \sum_{i\in[l]} v_i = H(\sum_{i\in[l]} h_{\hat{a}_i}(\hat{z}_i) - S \cdot v, \tilde{\beta}, A, \tilde{R}, \mu)$.

After receiving a l-ring public keys R associated with $\sigma_{\mathbb{B}}$, $\mathbb{B}$ forwards them to $\mathbb{A}$ as the public key for $\sigma_{\mathbb{A}}$. When $\mathbb{A}$ asks queries of Add, Reg, and Crpt, $\mathbb{B}$ simply passes these queries to its own challenger, and returns the received answers to $\mathbb{A}$. When $\mathbb{A}$ asks the DRSig query or the EGen query, $\mathbb{B}$ handles it by controlling the random oracle H and when $\mathbb{A}$ asks the hash H query with the entry $(x_1, x_2, x_3, x_4, x_5)$, $\mathbb{B}$ asks its own hash oracle with the entry $(x_1, x_4, x_2||x_3||x_5)$, and returns the answer to $\mathbb{A}$. $\mathbb{B}$ maintains the consistence of the H outputs. If an answer from its own hash oracle happens to be the same as the one $\mathbb{B}$ has already used to answer $\mathbb{A}$'s DRSig or EGen query before, $\mathbb{B}$ aborts. We can argue that since the size of the hash function H is reasonably large and $\mathbb{B}$ chooses every hash output randomly, so the probability of two hash queries hitting to the same output is negligible.

When $\mathbb{A}$ comes up with a forged signature $\sigma_{\mathbb{A}} = (\hat{b}, \tilde{z}_U, A, \tilde{v}_U)$ with $v = \sum_{i\in[l]} v_i = H(\sum_{i\in[l]} h_{\hat{a}_i}(\hat{z}_i) - S \cdot v, \tilde{\beta}, A, \tilde{R}, \mu)$, $\mathbb{B}$ submits its own forged signature as $\sigma_{\mathbb{B}} = (\tilde{z}_U, v)$ together with the signed message $\tilde{\beta}||A||\mu$.

The unforgeability of the Aguilar-Melchor et al. ring signature scheme in [10] has been proved under the random oracle model and their proof shows that the capability of forging a ring signature from their scheme can be used to solve $SVP_\gamma(\mathbb{L})$ for $\gamma = \tilde{O}(n^{2.5+2c})$ for every lattice $\mathbb{L}$ corresponding to an ideal $\mathbb{D}$, which is believed to be a computationally hard problem. Following the above discussion, we can claim that forging a ring signature from our scheme is also computationally infeasible. □

Lemma 2 *If there is a polynomial-time algorithm that can forge a ring signature from the NDRS scheme, there is another polynomial-time algorithm that can forge a signature from the Lyubashevsky signature scheme in* [19].

Proof Our *NDRS* scheme involves two connected ring signatures: one is based on the ring signature scheme in [10] as discussed in the previous lemma; and the other is based on the Lyubashevsky signature scheme in [19]. Recall that the Lyubashevsky signature scheme has the same public and private key setting as our *NDRS* scheme. Part of our *NDRS* scheme can be seen as the Lyubashevsky signature scheme. Suppose that the real signer of our *NDRS* scheme is i, its private is $\hat{s}_i$ and the corresponding ephemeral public key is $(\sigma_i, \hat{b})$ for $\sigma_i = h_{\hat{b}}(\hat{s}_i)$. If there is an adversary $\mathbb{A}$ who is able to forge a ring signature from our construction, there exists another adversary $\mathbb{B}$ who can play the role as $\mathbb{A}$'s challenger and use $\mathbb{A}$'s response to forge a signature of [19], provided that $\mathbb{B}$ is able to control the random oracle H used by $\mathbb{A}$.

Let $\sigma_{\mathbb{B}} = (\tilde{z}, e)$ be a signature from [19], where $e = H(h_{\hat{b}}(\hat{z}) - S^* \cdot e, \mu)$. Let $\sigma_{\mathbb{A}} = (\hat{b}, \tilde{z}_U, A, \tilde{v}_U)$ be a ring signature from our construction, where $v = \sum_{i\in[l]} v_i = H(\alpha, \beta_1, \ldots, \beta_{i-1}, h_{\hat{b}}(\hat{z}_i) - S^* \cdot v, \beta_{i+1}, \ldots, \beta_l, A, \tilde{R}, \mu)$. In this discussion, $S^* = \sigma_i$.

In the process of forging a Lyubashevsky signature, we allow $\mathbb{B}$ to update a signer's public key by using a fresh hash function $h_{\hat{b}}()$. It is easy to see that this extra power does not change the nature of the Lyubashevsky signature scheme.

After receiving a public keys $pk = \hat{a}$ associated with $\sigma_{\mathbb{B}}$, $\mathbb{B}$ sets $\hat{a}_i = \hat{a}$ as the i-th public key and selects other $l-1$ public keys in the *NDRS* scheme at random, and forwards them together to $\mathbb{A}$ as the public key for $\sigma_{\mathbb{A}}$. When $\mathbb{A}$ asks the queries of Add, Reg, and Crpt, $\mathbb{B}$ simply passes these queries to its own challenger, and returns the received answers to $\mathbb{A}$, but if $\mathbb{A}$ asks the Crpt query with the entry i, $\mathbb{B}$ has to abort. With the probability of $1/N$, i is used as the real signer and in this case $\mathbb{A}$ will not ask the Crpt query for i. When $\mathbb{A}$ asks the DRSig query or the EGen query, $\mathbb{B}$ handles it by controlling the random oracle H. When $\mathbb{A}$ asks the hash H query with the entry $(x_1, x_{21}, \ldots, x_{2i}, \ldots, x_{2l}, x_3, x_4, x_5)$, $\mathbb{B}$ asks its own hash oracle with the entry $(x_{2i}, x_1||x_{21}|| \cdots x_{2(i-1)}||x_{2(i+1)}|| \cdots x_{2l}||x_3||x_4||x_5)$, and returns the answer to $\mathbb{A}$. $\mathbb{B}$ maintains the consistence of the H outputs. If an answer from its own hash oracle happens to be the same as the one $\mathbb{B}$ has already used to answer $\mathbb{A}$'s DRSig or EGen query before, $\mathbb{B}$ aborts. We can argue that since the size of the hash function H is reasonably large and $\mathbb{B}$ chooses every hash output randomly, so the probability of two hash queries hitting to the same output is negligible.

When $\mathbb{A}$ comes up with a forged signature $\sigma_{\mathbb{A}} = (\hat{b}, \tilde{z}_U, A, \tilde{v}_U)$ with $v = \sum_{i\in[l]} v_i = H(\sum_{i\in[l]} h_{\hat{a}_i}(\hat{z}_i) - S \cdot v, \tilde{\beta}, A, \tilde{R}, \mu)$, $\mathbb{B}$ first asks its own challenger to update the i-th public key by replacing $\hat{a}_i$ with $\hat{b}$, and then submits its own forged signature as $\sigma_{\mathbb{B}} = (\tilde{z} = \beta_i, v)$ together with the signed message $\alpha||\beta_1|| \cdots ||\beta_{i-1}||\beta_{i+1}|| \cdots ||\beta_l||A||\tilde{R}||\mu$.

The unforgeability of Lyubashevsky's signature scheme in [19] has been proved under the random oracle model and their proof shows that the capability of forging a ring signature from their scheme can be used to solve $SVP_\gamma(\mathbb{L})$ for $\gamma = \tilde{O}(n^{2.5})$ for every lattice $\mathbb{L}$ corresponding to an ideal $\mathbb{D}$, which is believed to be a computationally hard problem. Following the above discussion, we can claim that forging a ring signature from our scheme is also computationally infeasible. □

Lemma 3 *If there is a polynomial-time algorithm that can forge a piece of evidence from the NDRS scheme, there is*

another polynomial-time algorithm that can forge a signature from the Lyubashevsky signature scheme in [19].

Proof A piece of evidence in our *NDRS* scheme can be seen as two signatures from the Lyubashevsky signature scheme in [19]. Suppose that the real signer of our *NDRS* scheme is i, its private key is $\hat{s}_i$ and the corresponding long-term public key is $(\hat{a}_i, S = h_{\hat{a}_i}(\hat{s}_i))$ and ephemeral public key is $(\hat{b}, \sigma_i = h_{\hat{b}}(\hat{s}_i))$. If there is an adversary $\mathbb{A}$ who is able to forge evidence of i from our construction, there exists another adversary $\mathbb{B}$ who can play the role as $\mathbb{A}$'s challenger and use $\mathbb{A}$'s response to forge a signature of [19], provided that $\mathbb{B}$ is able to control the random oracle H used by $\mathbb{A}$.

Following the same approach used in the proof of the previous lemma, in the process of forging a Lyubashevsky signature, we allow $\mathbb{B}$ to update a signer's public key by using a fresh hash function $h_{\hat{b}}()$. As a result, each signer i will have two public keys corresponding to a single secret key.

Let $\sigma_{\mathbb{B}} = (\tilde{z}, e)$ be a signature from [19], where $e = H(h_{\hat{b}}(\hat{z}) - S^* \cdot e, \mu)$. Let $\sigma_{\mathbb{A}} = (\sigma_i, \alpha_i, \beta_i, \hat{z}_i, e_i)$ be a piece of evidence of the *NDRS* scheme, where $e_i = H(\alpha_i, \beta_i, A, \tilde{R}, \mu)$ with $\alpha_i = h_{\hat{a}_i}(\hat{z}_i) - S \cdot e_i$ and $\beta_i = h_{\hat{b}}(\hat{z}_i) - \sigma_i \cdot e_i$.

After receiving a public keys $pk = \hat{a}$ associated with $\sigma_{\mathbb{B}}$, $\mathbb{B}$ sets $\hat{a}_i = \hat{a}$ as the i-th public key and asks its own challenger to update the i-th public key by replacing $\hat{a}_i$ with $\hat{b}$. $\mathbb{B}$ forwards $\hat{a}_i$ and $\hat{b}$ to $\mathbb{A}$. When $\mathbb{A}$ asks the queries of Add, Reg, and Crpt, $\mathbb{B}$ simply passes these queries to its own challenger, and returns the received answers to $\mathbb{A}$. When $\mathbb{A}$ asks the DRSig query or the EGen query, $\mathbb{B}$ handles it by controlling the random oracle H. When $\mathbb{A}$ asks the hash H query with the entry $(x_1, x_2, x_3, x_4, x_5)$, $\mathbb{B}$ asks its own hash oracle with the entry $(x_1, x_2||x_3||x_4||x_5)$ if it wants to forge a Lyubashevsky signature under pk_i or with the entry $(x_2, x_1||x_3||x_4||x_5)$ if it wants to forge a Lyubashevsky signature under σ_i, and returns the answer to $\mathbb{A}$. $\mathbb{B}$ maintains the consistence of the H outputs. If an answer from its own hash oracle happens to be the same as the one $\mathbb{B}$ has already used to answer $\mathbb{A}$'s DRSig or EGen query before, $\mathbb{B}$ aborts. We can argue that since the size of the hash function H is reasonably large and $\mathbb{B}$ chooses every hash output randomly, so the probability of two hash queries hitting to the same output is negligible.

When $\mathbb{A}$ comes up with a piece of forged evidence $\sigma_{\mathbb{A}} = (\sigma_i, \alpha_i, \beta_i, \hat{z}_i, e_i)$ with $\alpha_i = h_{\hat{a}_i}(\hat{z}_i) - S \cdot e_i$ and $\beta_i = h_{\hat{b}}(\hat{z}_i) - \sigma_i \cdot e_i$, $\mathbb{B}$ submits its own forged signature as $\sigma_{\mathbb{B}} = (\tilde{z}, e_i)$ together with the signed message $\beta_i||A||\tilde{R}||\mu$ if $(\hat{a}, S)$ was the target public key and $e_i = H(\alpha_i, \beta_i||A||R||\mu)$, or together with the signed message $\alpha||A||\tilde{R}||\mu$ if $(\hat{b}, \sigma_i)$ was the target public key and $e_i = H(\beta_i, \alpha||A||\tilde{R}||\mu)$.

Following the above discussion, we can claim that forging a piece of evidence from our scheme is also computationally infeasible. □

The theorem follows from the combination of these lemmas with the above discussion.

Theorem 4 (non-frameability) *Assume that the Aguilar-Melchor et al. ring signature scheme* [10] *and the Lyubashevsky signature scheme* [19] *are unforgeable and that finding a pre-image for the hash function $h_{\hat{b}}()$ is computationally infeasible, the proposed NDRS scheme described in Sect. 4 holds the property of non-frameability under the random oracle model.*

Proof Recall that a signature from the *NDRS* scheme includes two connected ring signatures, the first one is under the long-term public keys and the second one is under the ephemeral public keys. For the purpose of this proof, we call these two ring signatures the first and second partial ring signatures. As we have given a proof in Theorem 3, the first partial ring signature is unforgeable due to the unforgeability of the Aguilar-Melchor et al ring signature scheme (see Lemma 1), and the second partial ring signature is unforgeable due to the unforgeability of the Lyubashevsky signature scheme (see Lemma 2). This means the signer must make use of at least one private key corresponding with a public key that is included in the l-ring public keys $R = \{pk_1, \ldots, pk_l\}$ and at least one private key corresponding with an ephemeral public key in the l-ring public keys σ_i for $i \in \{1, \ldots, l\}$.

We have also given a proof that a piece of evidence from the *NDRS* scheme is unforgeable due to the unforgeability of the Lyubashevsky signature scheme (see Lemma 3), that means the signer must make use of the private key corresponding with a public key, again, which must be included in the l-ring public keys $R = \{pk_1, \ldots, pk_l\}$.

To prove the non-frameability, we have to answer the question whether it is possible that a ring signature created under the key j but is able to be linked with the evidence indicating the key i? Alternatively the question is whether it is possible, the first partial ring signature has the real signer j but the second partial ring signature has the real signer i, where $j \neq i$. Furthermore the adversary does not know the secret key of i, $\hat{s}_i$. If the answers to these two questions are positive, the adversary wins the game of the non-frameability.

Because the evidence is unforgeable, in order to make the EvidenceCheck algorithm output "confirmation" for the signer i, the adversary knowing the private key $sk_j = \hat{s}_j$ but not $sk_i = \hat{s}_i$ must successfully create a ring signature σ of the *NDRS* scheme (including both of these two partial ring signatures) with the values A and $\hat{b}$, satisfying the following equation:

$$H_1\left(i||\hat{a}_i\right) \cdot S + A = h_{\hat{b}}\left(\hat{s}_i\right) = \sigma_i \tag{1}$$

Obviously the adversary can find such A and $\hat{b}$ values if they are available from the existing ring signatures created by the signer i. However, without the knowledge of $\hat{s}_i$, since the adversary is not allowed to ask the Reg or Crpt query with the entry i, can the adversary insert such A and $\hat{b}$ values

Table 2 Comparison of security and functionality

Scheme	Quantum-resistance	Non-interactivity	Deniability
KM [8]	No	No	Yes
WS [18]	Yes	Yes	No
AM [10]	Yes	Yes	No
This work	Yes	Yes	Yes

Table 3 Comparison of key and signature sizes (in bits)

Scheme	Public key	Private key	Signature
WS [18]	$mn \log q$	$2m^2 \log(n_B)$	$2lm \log r$
AM [10]	$nm \log p$	$2mn$	$2lnm \log \rho + 2n$
This work	$nm \log p$	$2mn$	$2lnm \log \rho + 2ln + (m+1)n \log p$

into a ring signature created by itself? Following the security definition of the non-frameability property, this ring signature must not be a result of the DRSig query.

To use Eq. (1) to create a valid ring signature, the adversary must make use of $\forall j \neq i, \sigma_j = H_1(j||\hat{a}_j) \cdot S + A$. Can the adversary find a pre-image $\hat{x}$ satisfying $\sigma_j = h_{\hat{b}}(\hat{x})$? The answer is NO, since we assume that finding a pre-image for this hash function is computationally infeasible.

To summarize this discussion, we can argue that the adversary cannot create a valid ring signature satisfying Eq. (1) since otherwise, the adversary can either forge the second partial ring signature in the ring signature from the *NDRS* scheme which contradicts Lemma 2 or, they must be able to find a pre-image $\hat{x}$ of $h_{\hat{b}}(\hat{x})$ which contradicts the hash function assumption.

Without satisfying the condition in Eq. (1), the ring signature cannot be traced to i since the evidence is unforgeable. The theorem follows with this reasoning. □

6 Comparison with previous work

In this section, we compare our work on non-deniable ring signatures with the ring signature schemes from Komano et al. [8], Wang and Sun [18] and Aguilar-Melchor et al. [10] (these will be referred to as KM, WS and AM in what follows). Table 2 shows the comparison of this work with the other three works in terms of their security and functionality.

For the key and signature sizes we compare this work with the other two lattice-based schemes. First we provide a brief summary of the elements of each scheme. Here, l denotes the number of ring members used in the signature.

6.1 Wang and Sun [18]

Parameters m, n, q are positive integers with $q \geq 2$ and $m \geq 5n \log q$. Parameter r is a Gaussian parameter used to generate the secret basis and short vectors, which is defined by $r \geq \tilde{L} \cdot \omega(\sqrt{\log n})$ and $r \geq O(\sqrt{n \log q})$.

- *Public key* A matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ defines a lattice, $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = 0 \mod q\}$.
- *Secret key* A matrix $\mathbf{B} \in \mathbb{Z}^{m \times m}$, a short basis for the lattice $\Lambda^{\perp}(\mathbf{A})$ with $||\mathbf{B}|| \leq O(n \log q)$. For this comparison we take $||\mathbf{B}|| \leq Cn \log q$, for some constant C and set $n_B = Cn \log q / \sqrt{m}$.
- *Signature* The ring of public keys, $\mathbf{A}_R = \{\mathbf{A}_1, \ldots, \mathbf{A}_l\}$ with $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{e} \in \mathbb{Z}^{Nm}$ with $||\mathbf{e}|| \leq r\sqrt{lm}$.

6.2 Aguilar-Melchor et al. [10]

Parameters n, p, m in [10] are shown in Table 1. For simplicity, we write $\rho = mn^{1.5} \log n - \sqrt{n} \log n$.

- *Public key* A vector of polynomials, $\hat{a} \in \mathbb{D}^m$.
- *Secret key* A vector of polynomials, $\hat{s} \in \mathbb{D}_s^m$.
- *Signature* The ring of public keys, $\{\hat{a}_1, \ldots, \hat{a}_l\}$ with $\hat{a}_i \in \mathbb{D}^m$, a vector $\tilde{z}_U = (\hat{z}_1, \ldots, \hat{z}_l)$ with $\hat{z}_i \in \mathbb{D}_z^m$ and a polynomial, $v \in \mathbb{D}_s$.

6.3 Our scheme

The keys in our scheme are the same as in AM, but the addition of deniability results in larger signature sizes.

- *Signature* The ring of public keys, $\{\hat{a}_1, \ldots, \hat{a}_l\}$ with $\hat{a}_i \in \mathbb{D}^m$, a vector of polynomials, $\hat{b} \in \mathbb{D}^m$, a polynomial, $A \in \mathbb{D}$, a vector $\tilde{z}_U = (\hat{z}_1, \ldots, \hat{z}_l)$ with $\hat{z}_i \in \mathbb{D}_z^m$ and a vector of polynomials, $\tilde{v}_U = (v_1, \ldots, v_l)$ with $v_i \in \mathbb{D}_s$.

Table 3 shows the comparison of our work with other related work in terms of the sizes of their public keys, private keys and signatures.

As might be expected, this summary shows that our work has similar key sizes to the AM scheme so it is equally reasonable and feasible. In addition, our scheme is not only secure from quantum attack, but also with some cost in signature size to be non-interactive and deniable.

7 Conclusion and future work

In this paper, we proposed a lattice-based non interactive deniable ring signature scheme. A number of real-world scenarios were suggested to demonstrate possible applications of such a scheme. In order to cover the property of non-interaction, we employed the security model modified from the interactive deniable ring signature scheme introduced by Komano et al. [8]. We then proved the security of our proposed scheme under the modified security model. In our construction, the interactive confirmation/disavowal protocol in [8] was replaced by signature-based evidence generation and check algorithms, which can tell whether the evidence producer is the real signer or not. The security of our proposed scheme is based on the SVP_γ hard lattice problem, so the scheme is believed to be quantum-resistent.

Our signature size grows linearly with the number of the ring members, a property which it inherits from the scheme of Aguilar-Melchor et al. [10]. Libert et al. [20] proposed a lattice-based ring signature scheme with a signature size that is logarithmic in the cardinality of the ring. We leave the construction of a shorter deniable ring signature scheme for future exploration.

Acknowledgements This work is supported by the National Natural Science Foundations of China (Nos. 61472309, 61572390, and 61672412), and National Cryptography Development Fund under Grant (MMJJ20170104)

References

1. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret: theory and applications of ring signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science, Essays in Memory of Shimon Even, pp. 164–186. Springer, Berlin (2006). https://doi.org/10.1007/11685654_7
2. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Christian, C., Camenisch, J.L. (eds.) EUROCRYPT'04—Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Lecture Notes in Computer Science, vol. 3027, pp. 609–626. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-24676-3_36. http://link.springer.com/chapter/10.1007%2F978-3-540-24676-3_36
3. Naor, M.: Deniable ring authentication. In: Yung, M. (ed.) CRYPTO'02—Proceedings of 22nd Annual International Cryptology Conference, Lecture Notes in Computer Science, vol. 2442, pp. 481–498. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45708-9_31. http://link.springer.com/chapter/10.1007%2F3-540-45708-9_31
4. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) Advances in Cryptology—EUROCRYPT 1991, pp. 257–265. Springer, Berlin (1991). https://doi.org/10.1007/3-540-46416-6_22
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) Advances in Cryptology—EUROCRYPT 2003, pp. 614–629. Springer, Berlin (2003). https://doi.org/10.1007/3-540-39200-9_38
6. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) Advances in Cryptology—EUROCRYPT 2006, pp. 427–444. Springer, Berlin (2006). https://doi.org/10.1007/11761679_26
7. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) Advances in Cryptology—ASIACRYPT 2007, pp. 164–180. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-76900-2_10
8. Komano, Y., Ohta, K., Shimbo, A., Kawamura, S.: Toward the fair anonymous signatures: Deniable ring signatures. In: Pointcheval, D. (ed.) CT-RSA 2006—Topics in Cryptology, pp. 174–191. Springer, Berlin (2006). https://doi.org/10.1007/11605805_12
9. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172
10. Aguilar-Melchor, C., Bettaieb, S., Boyen, X., Fousse, L., Gaborit, P.: Adapting Lyubashevsky's signature schemes to the ring signature setting. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013—Progress in Cryptology: 6th International Conference on Cryptology in Africa, pp. 1–25. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-38553-7_1
11. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret, pp. 552–565. Springer, Berlin (2001). https://doi.org/10.1007/3-540-45682-1_32
12. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: the case of dynamic groups. In: Menezes, A. (ed.) CT-RAS 2005—Topics in Cryptology, pp. 136–153. Springer, Berlin (2005). https://doi.org/10.1007/978-3-540-30574-3_11
13. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC '96—Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM, New York (1996). https://doi.org/10.1145/237814.237838
14. Brakerski, Z., Kalai, Y.T.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology Eprint Archive, Report 2010/086 (2010). http://eprint.iacr.org/2010/086
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC '08—Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM , New York (2008). https://doi.org/10.1145/1374376.1374407
16. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010—Advances in Cryptology: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 523–552. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-13190-5_27

17. Wang, F.H., Hu, Y.P., Wang, C.X.: A lattice-based ring signature scheme from bonsai trees. J. Electron. Inf. Technol. **32**(10), 2400 (2010). https://doi.org/10.3724/SP.J.1146.2009.01491. http://jeit.ie.ac.cn/EN/abstract/article_14899.shtml
18. Wang, J., Sun, B.: Ring signature schemes from lattice basis delegation. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS 2011—Information and Communications Security, pp. 15–28. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-25243-3_2
19. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009—Advances in Cryptology: 15th International Conference on the Theory and Application of Cryptology and Information Security, pp. 598–616. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-10366-7_35
20. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology—EUROCRYPT 2016, pp. 1–31. Springer, Berlin (2016). https://doi.org/10.1007/978-3-662-49896-5_1
21. Yoo, Y., Azarderakhsh, R., Jalali, A.,Jao, D.,Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In: Kiayias, A. (eds.) FC 2017—Financial Cryptography and Data Security, pp. 163–181. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_9
22. Feo, L.D., Jao, D., Plû, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209C247 (2014). https://doi.org/10.1515/jmc-2012-0015
23. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006—Automata, Languages and Programming: 33rd International Colloquium, Part II, pp. 144–155. Springer, Berlin (2006). https://doi.org/10.1007/11787006_13
24. Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems: a cryptographic perspective. The Kluwer International Series in Engineering and Computer Science, vol. 671. Kluwer Academic Publishers, Boston (2002). https://doi.org/10.1007/978-1-4615-0897-7
25. Sweeney, L.: k-Anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
26. Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M.: l-Diversity: privacy beyond k-anonymity. In: ACM Transactions on Knowledge Discovery from Data, vol. 1, No. 1, Article 3 (2007)
27. Lyubashevsky, V.: Towards practical lattice-based cryptography. In: Ph.D. Thesis, University of California, San Diego (2008)