DualRing: Generic Construction of Ring Signatures with Efficient Instantiations[⋆]

Tsz Hon Yuen[1], Muhammed F. Esgin[2,3], Joseph K. Liu[2], Man Ho Au[1], and Zhimin Ding[4]

[1] The University of Hong Kong, Hong Kong
{thyuen, allenau}@cs.hku.hk
[2] Department of Software Systems and Cybersecurity,
Faculty of Information Technology, Monash University, Australia
{muhammed.esgin, joseph.liu}@monash.edu
[3] CSIRO's Data61, Australia
[4] Rice University, USA
zd21@rice.edu

Abstract. We introduce a novel generic ring signature construction, called *DualRing*, which can be built from several canonical identification schemes (such as Schnorr identification). *DualRing* differs from the classical ring signatures by its formation of *two* rings: a ring of commitments and a ring of challenges. It has a structural difference from the common ring signature approaches based on accumulators or zero-knowledge proofs of the signer index. Comparatively, *DualRing* has a number of unique advantages.

Considering the DL-based setting by using Schnorr identification scheme, our *DualRing* structure allows the signature size to be compressed into logarithmic size via an argument of knowledge system such as Bulletproofs. We further improve on the Bulletproofs argument system to eliminate about half of the computation while maintaining the same proof size. We call this *Sum Argument* and it can be of independent interest. This DL-based construction, named DualRing-EC, using Schnorr identification with *Sum Argument* has the shortest ring signature size in the literature without using trusted setup.

Considering the lattice-based setting, we instantiate *DualRing* by a canonical identification based on M-LWE and M-SIS. In practice, we achieve the shortest lattice-based ring signature, named DualRing-LB, when the ring size is between 4 and 2000. DualRing-LB is also 5× faster in signing and verification than the fastest lattice-based scheme by Esgin et al. (CRYPTO'19).

Keywords: Ring Signature · Generic Construction · Sum Argument · M-LWE/SIS

1 Introduction

Ring signatures [37] allow a signer to dynamically choose a set of public keys (including his/her own) and to sign messages on behalf of the set, without revealing who the real signer is. In addition, it is impossible to check if two signatures are issued by the same signer. Ring signatures provide anonymity and they are widely used in privacy-preserving protocols such as e-voting, whistleblowing and privacy-preserving cryptocurrencies.

Classical Ring Structure. The classical ring signatures [37] for a set of n public keys $\mathbf{pk}$ are constructed by computing $n-1$ "pseudo-signatures" (the outputs computed from the verification function) sequentially in a *ring structure* first and then using one signer secret key to create a real signature. These n signatures together form a ring signature on behalf of $\mathbf{pk}$.

Abe *et al.* [2] generalized this idea in a generic construction (AOS ring signature), which can be built from two types of standard signatures: Type-H (Hash-and-one-way type, e.g., RSA signature) and Type-T (Three-move type, e.g., Schnorr signature). Borromean ring signatures [35] used the ring structure in [2] to compress multiple ring signatures. Its variant is used in privacy-preserving cryptocurrency Monero.

[⋆] This is the full version of the paper in [38].

From Accumulator to Zero-Knowledge Proof. The major drawback of the above ring structure approach is the signature size of $O(n)$. Therefore, researchers used other cryptographic primitives to build ring signatures.

An accumulator allows the signer to "compress" n public keys into a constant size *value* and there is a *witness* showing that the signer's public key is in the set of public keys. The advantage of the accumulator-based ring signature [17] is the constant signature size. However, most of the existing accumulators require a trusted setup, which is often not desirable.

Another main approach to constructing an efficient ring signature is to use a zero-knowledge proof to prove knowledge of the secret key with respect to one of the public keys in the ring. The state-of-the-art proof size is $O(\log n)$ by the use of one-out-of-many proof [23].

1.1 DualRing: New Generic Construction of Ring Signature

In this paper, we revisit the classical ring structure approach and design a novel *dual ring* structure to build a new generic construction of ring signatures. Let us first recall how a Type-T signature works and how the AOS ring signature [2] is built on top of it.

A Type-T signature involves the following three functions in its signing (we use Schnorr signature as a running example, indicated inside [ ], with a secret key sk, a public key $\mathsf{pk} = g^{\mathsf{sk}}$ and a message M): a commit function A, which outputs a commitment R [$A : g^r \to R$]; a hash function H, which outputs a challenge c [$H(M, R) \to c$]; and a response function Z, which outputs a response z [$Z : r - c \cdot \mathsf{sk} \to z$]. A Type-T signature is then $\sigma = (c, z)$. For the verification algorithm, one runs a function V to reconstruct R from σ [$V : g^z \cdot \mathsf{pk}^c \to R'$], and then runs H to check if c is correct [$H(M, R') \overset{?}{=} c$].

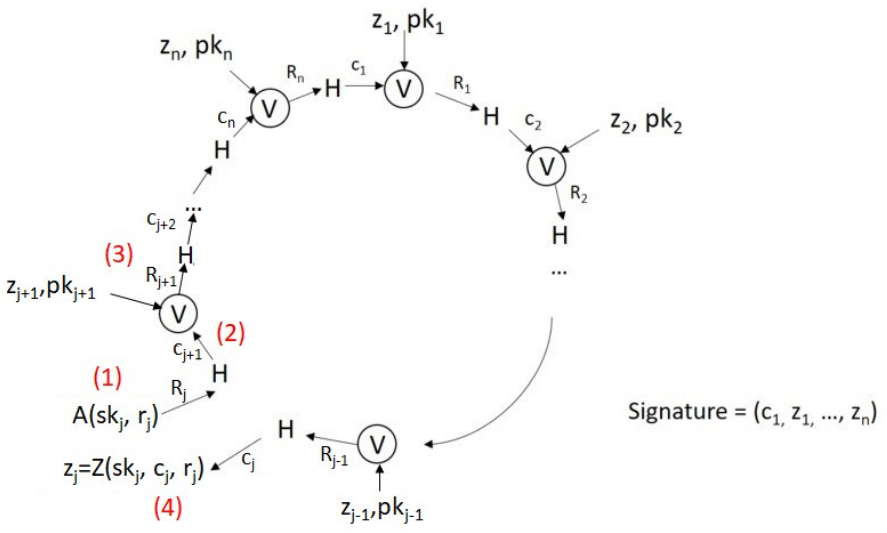


Fig. 1: Structure of the AOS ring signature from a Type-T Signature in [2].

Now, in a Type-T AOS *ring* signature for public keys $\mathbf{pk} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, the signer (with index j) follows the structure in Fig. 1, where the signer is assumed to have sk_j corresponding to pk_j. In particular, (1) the signer picks a randomness r_j to generate R_j via the commit function A. (2) The signer uses the commitment R_j to compute the $(j+1)$-th challenge c_{j+1} by the hash function H. (3) For $i = j+1, \ldots, n, 1, \ldots, j-1$ by picking a random $(i+1)$-th response z_i and the public key of the (i)-th user pk_i, the signer can reconstruct the (i)-th commitment R_i using the function V as in verification and generate the $(i+1)$-th challenge c_{i+1} by the hash function H. A *ring* is then formed sequentially. (4) The last step is to compute z_j from sk_j, c_j, r_j using the

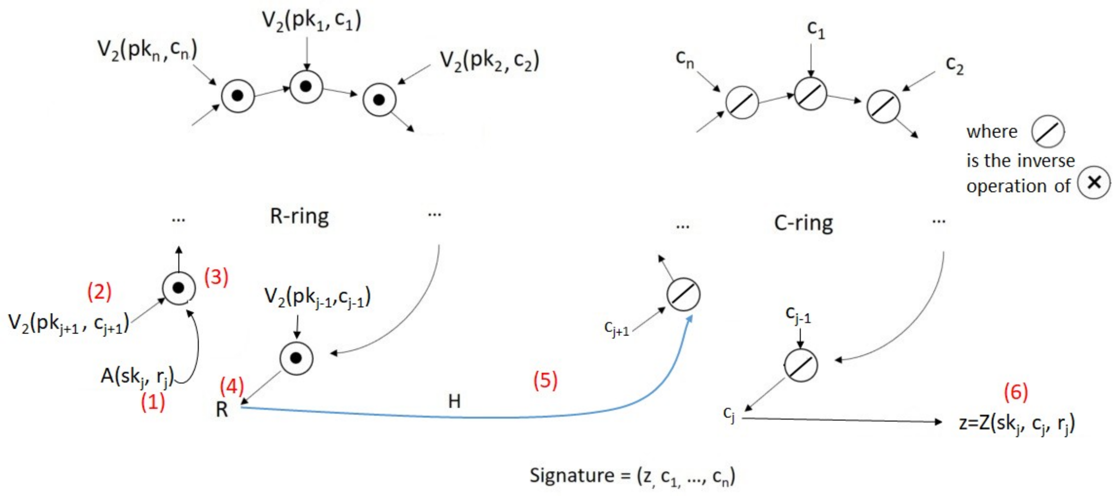


Fig. 2: Structure of DualRing construction

response function Z. The final ring signature is composed of a single challenge c_1 and n responses $(z_1, \ldots, z_n)$.

Overview of DualRing. We now describe our novel generic construction of ring signatures called *DualRing*. Let $\odot$ and $\otimes$ be two commutative group operations (e.g., modular multiplication and modular addition). We first modify the definition of a Type-T signature as follows:

- the verification function $V(\mathsf{pk}, z, c)$ within the verification algorithm can be divided into two functions $V_1(z)$ and $V_2(\mathsf{pk}, c)$ (pk is the public key, c is the challenge and z is the response) such that

$$V(\mathsf{pk}, z, c) = V_1(z) \odot V_2(\mathsf{pk}, c) \qquad \text{[Schnorr: } V_1 : g^z, V_2 : \mathsf{pk}^c\text{]}.$$

Using this property, we construct a ring signature with a dual-ring structure as in Fig. 2. Particularly, for a set of public keys $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and a secret key sk_j, (1) the signer first picks some randomness r_j. (2) He further picks random challenges $c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_n$, and (3) forms an R-ring using the group operation $\odot$ with the functions A and V_2. (4) Then he computes R as:

$$R = A(\mathsf{sk}_j; r_j) \odot V_2(\mathsf{pk}_{j+1}, c_{j+1}) \odot \cdots \odot V_2(\mathsf{pk}_n, c_n) \odot V_2(\mathsf{pk}_1, c_1) \odot \cdots \odot V_2(\mathsf{pk}_{j-1}, c_{j-1}).$$

After that, the signer forms a C-ring using the group operation $\otimes$, where the “missing” challenge (5) c_j is computed as:

$$c_j = H(M, \mathbf{pk}, R) \oslash c_{j+1} \oslash \cdots \oslash c_n \oslash c_1 \oslash \cdots c_{j-1} \text{ (where } \oslash \text{ is the inverse of } \otimes\text{)}.$$

As a result, the following equation is satisfied

$$c_1 \otimes \cdots \otimes c_n = H(M, \mathbf{pk}, R) \tag{1}$$

to form the link connecting the two rings for the input message M and the list of public key $\mathbf{pk}$. (6) Lastly, the response z is computed by running $Z(\mathsf{sk}_j, c_j, r_j)$. The final ring signature is composed of a *single response* z and n challenges $(c_1, \ldots, c_n)$, in contrast of the AOS signature which is composed of a *single challenge* c_1 and n responses $(z_1, \ldots z_n)$.

Advantages of DualRing over the AOS Ring Signature. The advantage of DualRing is threefold. Firstly, the AOS ring signature is composed of a *single* challenge and n responses, while

DualRing is composed of n challenges and a *single* response. When instantiated with cryptosystems having a small challenge size and a large response size (e.g., lattice-based cryptosystem), it leads to a significant saving in terms of signature size.

Secondly, we observe that the AOS ring signature includes the hash function H in the ring structure (Fig. 1), and this makes it difficult to further shorten the signature. On the other hand, DualRing uses two separate rings with simple group operations, which allows the use of an argument of knowledge to *efficiently* prove the relation in Eq. (1). We instantiate this in the discrete logarithm (DL) setting with communication complexity $O(\log n)$.

Thirdly, our DualRing, when instantiated with the Schnorr identification, has a simpler security reduction when compared to the alternative construction of the AOS ring signature in the Appendix A of [2]. They described that "*the reduction is quite costly because we may have to have at most n successful rewinding simulations*" and hence they did not give a full proof. On the other hand, our instantiation does not incur such security loss.

Technical Challenges. One of technical challenges we solve in this paper is to give a security proof for DualRing, as well as the $\mathsf{Type\text{-}T}$ AOS ring signature which has not been formally proven. Note that it has been an open problem to prove the security of the generic construction of the $\mathsf{Type\text{-}T}$ AOS ring signature [2] (only a security proof for the instantiation using the Schnorr signature was previously given). We solve this open problem by using *canonical identification* [1] (which is a three-move identification scheme that can be transformed to a $\mathsf{Type\text{-}T}$ *signature* by the Fiat-Shamir heuristic) in the construction and the security proofs. While the $\mathsf{Type\text{-}T}$ signature restricts the input to the hash function to include the *signer's public key*, the hash function H of the AOS ring signature takes the set of public keys $\mathbf{pk}$ as an input. This difference hinders the use of a forgery of the AOS ring signature to break the unforegability of the $\mathsf{Type\text{-}T}$ signature. On the other hand, the canonical identification does not have such a restriction on the generation of the challenge. The security proof of the $\mathsf{Type\text{-}T}$ AOS ring signature is given in the Appendix A.

In order to prove the security of DualRing, we further define a variant called $\mathsf{Type\text{-}T^*}$ *canonical identification*, with the following properties:

1. the verification $V(\mathsf{pk}, z, c)$ can be divided into two algorithms $V_1(z)$ and $V_2(\mathsf{pk}, c)$ such that $V(\mathsf{pk}, z, c) = V_1(z) \odot V_2(\mathsf{pk}, c)$;
2. V_1 is additively/multiplicatively homomorphic;
3. given the secret key sk corresponding to pk and a challenge c, there exists a function $\mathcal{T}$ which outputs $\hat{z} = \mathcal{T}(\mathsf{sk}, c)$ such that $V_1(\hat{z}) = V_2(\mathsf{pk}, c)$;
4. the challenge space Δ_c is a group.

Property 1 of $\mathsf{Type\text{-}T^*}$ canonical identification allows us to build the R-ring as in Fig. 2. Looking ahead, Property 3 is needed in the proof of DualRing's unforgeability to calculate $\hat{z}_i$ such that $V_1(\hat{z}_i) = V_2(\mathsf{pk}_i, c_i)$ for $i \neq j$, and then we use Property 2 to combine z with all $\hat{z}_i$'s to break the $\mathsf{Type\text{-}T^*}$ canonical identification. Property 4 is needed in the proof of DualRing's anonymity to make sure that the challenge c_j constructed in a specific way is indistinguishable from the others. We further define a new security model for canonical identification called *special impersonation*, which is a combination of the security models of impersonation and special soundness. Some standard identification schemes such as Schnorr identification and GQ identification [24] are examples of $\mathsf{Type\text{-}T^*}$ canonical identification secure against special impersonation.

1.2 Efficient Instantiations of DualRing

DualRing-EC: Logarithmic DL-based Ring Signature by Sum Argument. Having established a secure generic construction, DualRing, we try to compress the n challenges $(c_1, \dots c_n)$ via an argument of knowledge by exploiting the following simple algebraic structure:

$$c_1 \otimes \cdots \otimes c_n = H(M, \mathbf{pk}, R).$$

Ring Signatures	# elements in signature		Signature Size (Bytes)				
	$\mathbb{G}$	$\mathbb{Z}_p$	$n=2$	$n=8$	$n=64$	$n=2048$	$n=4096$
[30]	$4\log n+2$	$5\log n+4$	480	1070	1946	3114	3406
[23]	$4\log n$	$3\log n+1$	260	716	1400	2540	2768
[11]	$\log n+12$	$\frac{3}{2}\log n+6$	669	831	1074	1479	1560
[39]	$2\log n+7$	7	521	653	851	1181	1247
[28]	$2\log(n+2)+4$	5	424	523	721	1051	1117
DualRing-EC	$2\log n+1$	3	195	327	525	855	921

Table 1: $O(\log n)$-size DL-based ring signature schemes for n public keys, where p is a 256-bit prime.

This is theoretically a new approach to construct efficient ring signatures by combining the classical ring structure approach with the argument of knowledge[5].

In the DL setting, the group operation $\otimes$ is the modular addition. We improve the Bulletproof's inner product argument [14] into a new proof system called *Sum Argument*, which allows a prover to convince a verifier that he/she has the knowledge of a vector of scalars $(c_1,\ldots,c_n)$ such that their summation is a public value (i.e., $H(M,\mathbf{pk},R)$). Our *Sum Argument* only requires about half of the computation of Bulletproof while keeping the same proof size. We show how to obtain it by removing one of the two vectors of the inner product argument required in Bulletproof and to achieve a proof of size $O(\log n)$.

Based on DualRing, Schnorr identification and the sum argument above, we design DualRing-EC, the shortest ring signature scheme in the literature without using trusted setup, as shown in Table 1. The signature size is $O(\log(n))$. When implemented on an elliptic curve with a 256-bit modulus, DualRing-EC is at least 54% (resp., 27%, 18%) shorter than [28] for a ring size of 2 (resp. 64, 4096). Our scheme is at least 46% (resp., 64%, 67%) shorter than [30] for a ring size of 2 (resp. 64, 4096) at the same security level of 128-bit. Therefore, DualRing-EC is highly efficient and is useful for real world applications.

DualRing-LB: Shortest Lattice-based Ring Signature for Ring Size between 4 and 2000. We instantiate DualRing in the M-LWE/SIS setting and obtain DualRing-LB, the shortest lattice-based ring signature for a ring size between 4 and 2000. As mentioned above, DualRing-LB consists of a *single* response and n challenges. The size of a challenge (around 256 bits) in lattice-based identification is often much smaller than the size of a response (around a few KB). As a result, we obtain a compact lattice-based ring signature even without requiring a lattice-based sum argument. We compare with the shortest linear-size ring signature in [31] and shortest logarithmic-size ring signatures in [10,22] in Table 2. DualRing-LB is shorter than [10,22] for ring size less than about 2000 (note that our ring size can be arbitrary number). [31] is longer for all the ring sizes larger than 4, and it is based on a stronger NTRU assumption. The isogeny-based construction in [10] is at a much lower security level (60 bits of quantum security), is extremely slow (in the order of minutes), and has longer signatures than ours in the range around 5-300.

It is estimated in [19] that the running time of [19] is faster than Raptor for medium/large-sized rings ($n \geq 1024$) and also the estimated runtimes of [19] are significantly faster than those in [10]. The construction in [22] is an optimized version of that in [19] to reduce the signature length at the cost of computational efficiency. Therefore, the scheme by Esgin et al. [19] is the fastest scalable ring signature from lattices. We implement DualRing-LB together with the scheme in [19] and find that our scheme is at least 5 times faster in terms of sign and verify. We, therefore, expect an optimized implementation of our scheme to run faster than Raptor [31] and Falafl [10] as well for most ring sizes.

[5] Here, we do not require the zero-knowledge property since the anonymity of DualRing is provided by the ring structure.

Ring Signatures	Signature Size (Bytes)						Assumption
	$n = 2$	$n = 8$	$n = 64$	$n = 1024$	$n = 2048$	$n = 4096$	
Raptor [31]	2532	10128	81024	1296384	2592768	6564888576	NTRU
Falafl (for 2) [10]	49000	50000	52000	54000	54500	55000	M-LWE+M-SIS
MatRiCT [22]	18000	19000	31000	48000	53000	59000	M-LWE+M-SIS
DualRing-LB (Algo. 3 + 6)	4480	4630	6020	31160	55500	106570	M-LWE+M-SIS

Table 2: Lattice-based ring signatures for n public keys. DualRing-LB can be slightly optimized as described in Appendix D.

1.3 Our Contributions

Our contributions can be summarized as follows.

- The main contribution of our paper is the introduction of the novel dual ring structure *DualRing* to design generic construction of ring signatures, which differs significantly from the mainstream zero-knowledge-based or accumulator-based approaches.
- DualRing consists of n challenges and a single response, while the AOS ring signature consists of a single challenge and n responses. This significant difference allows us to produce much shorter signatures in both DL-based and lattice-based setting.
- In the DL-based setting, the DualRing structure allows the signature size to be compressed into $O(\log n)$ size, where n is the number of users in the ring, by using argument of knowledge system such as Bulletproofs [14]. We further enhance the Bulletproofs by eliminating almost half of the computation while maintaining the same proof size and thus achieve much better efficiency. We call this new argument of knowledge *Sum Argument* which can be of independent interest. Our resulting DualRing-EC deploying Schnorr identification scheme with *Sum Argument* is the shortest ring signature in the literature without using trusted setup.
- In the lattice-based setting, we instantiate DualRing by constructing a canonical identification based on M-LWE and M-SIS assumptions. DualRing-LB is the shortest lattice-based ring signature for the most practical ring sizes of 4 up to 2000.[6] We also implement DualRing-LB and show that it is at least 5 times faster in signing and verification than the state-of-the-art fastest construction (in terms of running times of signing and verification) in [19].

2 Related Work

Accumulator-Based Approach. Ring signatures can be constructed by accumulators [17]. The advantage of the accumulator approach is the constant signature size. However, the existing RSA-based and pairing-based accumulators both require a trusted setup for generating system parameters, which is not desirable for systems without a mutually trusted party. There exists a lattice-based accumulator [29] with no trusted setup, but it is not practical (the signature size is in the order of several MBs). Merkle-tree based accumulator does not require trusted setup. However, the membership proof of Merkle-tree based accumulator involves expensive zero-knowledge proof on hash function input.

Zero-Knowledge Proof Based Approach. The mainstream approach to construct a ring signature is to use a zero-knowledge proof on a signer index with the corresponding secret key.

[6] A ring signature of n users has some inherent limitations such that it requires at least n operations in signing and verification and storage of n public keys. These two limitations restrict the ring size to go up a lot for many practical applications. On the other hand, for very small ring sizes of, say, 2-5, the anonymity guarantee is very weak. For example, there has been attacks against Monero (cf. [27, 36]) that exploit the earlier use of very small rings of size < 6. Hence, one may argue that the most relevant range in practice falls inside 10-2000.

Most efficient schemes in the literature is to design a specific zero-knowledge proof for the designated cryptosystem (e.g., DL-based, RSA-based or lattice-based). In particular, a one-out-of-many proof [23] shows that the prover knows an opening of one out of n commitments. The index of such commitment can be expressed as a binary string $(b_1, \ldots b_{\log n})$. The zero-knowledge proof demonstrates the correctness of such an index, and hence the proof size is $O(\log n)$. Since a public key can be viewed as a commitment to zero[7], there are multiple ring signature schemes proposed using one-out-of-many proofs, including the DL-based setting [11,23] and lattice-based setting [19,20,22]. These ring signatures have size of $O(\log n)$.

Logarithmic-Size Generic Construction. The logarithmic-size generic construction of ring signature in [3] is secure in the standard model by using a public key encryption, a standard signature, a somewhere perfectly binding hash function with private local opening, and a non-interactive witness-indistinguishable (NIWI) proof systems. Their DL-based construction has a signature size of $2(\log n)^2 + 4$ elements in $\mathbb{G}$ and $2 \log n$ elements in $\mathbb{Z}_p$ with an additional NIWI proof (not instantiated in [3]), and hence it is not as efficient as the schemes in table 1. The lattice-based construction in [3] is also not efficient.

Recent and Parallel Work. A recent and parallel work by Lyubashevsky *et al.* [34] proposed a lattice-based ring signature scheme, which is logarithmic-size. Another concurrent work by Esgin *et al.* [21] introduced new techniques to instantiate the logarithmic-size ring signature in [22] more efficiently. Despite being linear-sized, our DualRing-LB remains smaller than these two recent proposals for about $n \leq 400$.

3 Preliminaries

Notations. In this paper, we use λ as the security parameter. For the notion $a \leftarrow_s S$, it means that we randomly pick an element a from a set S. We use bold letters such as $\mathbf{a}$ to represent a vector (or matrix for lattice-based construction).

Argument of Knowledge. An argument consists of three PPT algorithms $(\mathsf{S}, \mathcal{P}, \mathcal{V})$, which are CRS (Common Reference String) generator S, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. A CRS $\hat{\sigma}$ is produced by S on input λ and a transcript tr is produced by $\mathcal{P}$ and $\mathcal{V}$ on inputs s and t, which is denoted by $\mathsf{tr} \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ to denote that the verifier $\mathcal{V}$ accepts $b = 1$ or rejects $b = 0$. We define the language:

$$\mathcal{L} = \{x \mid \exists w : (\hat{\sigma}, x, w) \in \mathcal{R}\},$$

where w is a witness and x is a set of statements u in the relation $\mathcal{R}$.

An argument of knowledge $(\mathsf{S}, \mathcal{P}, \mathcal{V})$ should satisfy *perfect completeness* and *statistical witness-extended emulation* [12]. Informally, completeness means that a prover with a witness w for $x \in \mathcal{L}$ can convince the verifier of this fact. Statistical witness-extended emulation means that given an adversary that produces an acceptable argument with probability ϵ, there exists an emulator that produces a similar argument with probability ϵ together with a witness w.

Definition 1 (Perfect completeness). *For any non-uniform polynomial time adversary $\mathcal{A}$, $(\mathsf{S}, \mathcal{P}, \mathcal{V})$ has perfect completeness if*

$$\Pr\left[(\hat{\sigma}, u, w) \notin \mathcal{R} \text{ or } \langle \mathcal{P}(\hat{\sigma}, u, w), \mathcal{V}(\sigma, u) \rangle = 1 \,\middle|\, \hat{\sigma} \leftarrow \mathsf{S}(\lambda), (u, w) \leftarrow \mathcal{A}(\hat{\sigma}) \right] = 1.$$

Definition 2 (Statistical Witness-Extended Emulation). *For any deterministic polynomial time prover $\mathcal{P}^*$, $(\mathsf{S}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if there is a polynomial time emulator $\mathcal{E}$ such that for any pair of interactive adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ such that*

$$\Pr\left[\begin{array}{c|l} \mathcal{A}_1(\mathsf{tr}) & \hat{\sigma} \leftarrow \mathsf{S}(\lambda), \\ = 1 & (u, s) \leftarrow \mathcal{A}_2(\hat{\sigma}), \\ & \mathsf{tr} \leftarrow \langle \mathcal{P}^*(\hat{\sigma}, u, s), \\ & \quad \mathcal{V}(\hat{\sigma}, u) \rangle \end{array}\right] \approx \Pr\left[\begin{array}{l|c} \mathcal{A}_1(\mathsf{tr}) = 1 \wedge & \hat{\sigma} \leftarrow \mathsf{S}(\lambda), \\ (\mathsf{tr} \textit{ is accepting} & (u, s) \leftarrow \mathcal{A}_2(\hat{\sigma}), \\ \quad \Rightarrow (\hat{\sigma}, u, w) \in \mathcal{R}) & (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathcal{O}}(\hat{\sigma}, u) \end{array}\right],$$

[7] E.g., a DL-based public key g^x is a Pedersen commitment to zero.

where the oracle $\mathcal{O} = \langle \mathcal{P}^*(\hat{\sigma}, u, s), \mathcal{V}(\hat{\sigma}, u)\rangle$ can rewind to some point and resume with new randomness for the verifier $\mathcal{V}$ from this point onward.

Such an emulation above is used to define knowledge-soundness [12]. We consider s (which is the output of the adversary A_2 in the above equation) as the internal state of $\mathcal{P}^*$ with randomness, which follows that $\mathcal{E}$ can extract a witness whenever $\mathcal{P}^*$ generates a convincing argument in s.

4 Security Model

We review the security model of a ring signature in [8]. A ring signature consists of four PPT algorithms as follows:

$$\text{RS}\begin{cases}\mathsf{Setup}(\lambda) & \to \mathsf{param}\\ \mathsf{KeyGen}(\mathsf{param}) & \to (\mathsf{pk}, \mathsf{sk})\\ \mathsf{Sign}(\mathsf{param}, M, \mathbf{pk}, \mathsf{sk}) & \to \sigma\\ \mathsf{Verify}(\mathsf{param}, M, \mathbf{pk}, \sigma) & \to 1/0\end{cases}$$

We use $\mathbf{pk}$ to represent a vector of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$. For simplicity, we omit the input of system parameters param to algorithms other than Setup in the rest of this paper.

Unforgeability w.r.t. insider corruption. Unforgeability w.r.t. insider corruption [8] means that the adversary $\mathcal{A}$ cannot generate a valid signature without a secret key, even if he can adaptively corrupt some honest participants and obtain their secret keys.

Definition 3 (Unforgeability w.r.t. Insider Corruption). *For any polynomial time adversary $\mathcal{A}$, a ring signature is unforgeable if for some integer q_k polynomial in λ:*

$$\Pr\left[\begin{array}{l|l} 1 \leftarrow \mathsf{Verify}(M^*, \mathbf{pk}^*, \sigma^*), & \mathsf{param} \leftarrow \mathsf{Setup}(\lambda),\ \textit{for } i \in [1, q_k]: \\ \mathbf{pk}^* \subseteq S \setminus C, (M^*, \mathbf{pk}^*, \cdot) & (\hat{\mathsf{pk}}_i, \hat{\mathsf{sk}}_i) \leftarrow \mathsf{KeyGen}(), S := \{\hat{\mathsf{pk}}_i\}_{i=1}^{q_k}, \\ \textit{was not the input of } \mathcal{SO} & (M^*, \mathbf{pk}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{CO}, \mathcal{SO}}(\mathsf{param}, S) \end{array}\right] \le \mathsf{negl}(\lambda),$$

where the oracles given to $\mathcal{A}$ is defined as:

- *$\mathcal{CO}(i)$ outputs $\hat{\mathsf{sk}}_i$. We denote C as the set of corrupted users queried in $\mathcal{CO}$.*
- *$\mathcal{SO}(M, \mathbf{pk}, j)$: On input a message M, a vector of public keys $\mathbf{pk}$ and the signer index j, the Signing Oracle outputs $\perp$ if $\hat{\mathsf{pk}}_j \notin \mathbf{pk}$. Otherwise, it outputs a signature $\sigma \leftarrow \mathsf{Sign}(M, \mathbf{pk}, \hat{\mathsf{sk}}_j)$.*

Anonymity against full key exposure. We use the strong anonymity model in [8] that the adversary $\mathcal{A}$ is given all randomness to generate the secret keys.

Definition 4 (Anonymity against Full Key Exposure). *For any polynomial time adversary $(\mathcal{A}_1, \mathcal{A}_2)$, a ring signature is anonymous if for some integer q_k polynomial in λ:*

$$\left|\Pr\left[\begin{array}{l|l} & \mathsf{param} \leftarrow \mathsf{Setup}(\lambda),\ \textit{for } i \in [1, q_k]: \\ b = b', & (\hat{\mathsf{pk}}_i, \hat{\mathsf{sk}}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param}; \omega_i), \\ \hat{\mathsf{pk}}_{i_0}, \hat{\mathsf{pk}}_{i_1} & S := \{\hat{\mathsf{pk}}_i\}_{i=1}^{q_k}, \\ \in S \cap \mathbf{pk}^*. & (M^*, \mathbf{pk}^*, i_0, i_1, St) \leftarrow \mathcal{A}_1^{\mathcal{SO}}(\mathsf{param}, S), \\ & b \leftarrow_s \{0,1\}, \sigma \leftarrow \mathsf{Sign}(M^*, \mathbf{pk}^*, \hat{\mathsf{sk}}_{i_b}), \\ & b' \leftarrow \mathcal{A}_2(\sigma, \{\omega_i\}_{i=1}^{q_k}, St) \end{array}\right] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda).$$

Note that the set of public keys $\mathbf{pk}^*$ chosen by $\mathcal{A}_1$ can include adversarially generated public keys.

5 DualRing: Generic Ring Signature Construction

In this section, we show how to construct a generic ring signature scheme, DualRing, from a special kind of canonical identification scheme.

Algorithm 1: Type-T Signature

```
1  Procedure Setup($\lambda$):
2      define $H : \{0,1\}^* \to \Delta_c$;
3      return param;                  // including $H$
4  Procedure Sign($M$, sk):
5      $r \leftarrow \Delta_r$;
6      $R = A(\mathsf{sk}; r)$;
7      $c = H(M, R)$;
8      $z = Z(\mathsf{sk}, r, c)$;
9      return $\sigma = (z, c)$;
10 Procedure KeyGen():
11     return (pk, sk);
12 Procedure Verify($M$, pk, $\sigma$):
13     parse $\sigma = (z, c)$;
14     $R' = V(\mathsf{pk}, z, c)$;
15     if $c \neq H(M, R')$ then
16         return 0;
17     return 1;
```

Algorithm 2: Canonical Identification

```
1  Procedure Setup($\lambda$):
2      return param;
3  Procedure KeyGen():
4      return (pk, sk);
5  Procedure Proof1(sk):
6      $r \leftarrow_s \Delta_r$;
7      $R = A(\mathsf{sk}; r)$;
8      return $(R, r)$;
9  Procedure Ch($R$):
10     return $c$;
11 Procedure Proof2(sk, $r$, $c$):
12     return $z = Z(\mathsf{sk}, r, c)$;
13 Procedure Verify(pk, $z$, $c$):
14     $R' = V(\mathsf{pk}, c, z)$;
15     if $c \neq \mathrm{Ch}(R')$ then
16         return 0;
17     auxiliary checking with $R', c, z$;
18     return 1;
```

5.1 AOS Ring Signature

The AOS ring signature [2] can be constructed from a standard signature of Type-H or Type-T. We review the definition of Type-T in Algorithm 1.

- The Sign algorithm uses the algorithm A to generate a commitment R using a randomness r (chosen from a randomness domain Δ_r). Then, the message and R are hashed by H to obtain the hash value c (within the range of hash function Δ_c). Finally, the algorithm uses the function Z to generate the signature using the secret key sk, r and c.
- The Verify algorithm allows the reconstruction of R' from the public key pk, z and c using the function V. The signature is validated by using H on the message and R'.

Schnorr signature, Guillou-Quisquater signature [24], Katz-Wang signature [25] and EdDSA [9] are examples of Type-T signatures. Using Type-T signatures, a Type-T AOS ring signature can be constructed as shown in Fig. 1. However, as mentioned before, there is no security proof for this generic construction in [2], but only the instantiation with Schnorr signature is proven secure in [2]. In this paper, we give the *first* security proof for Type-T AOS ring signature in the Appendix A, which solves the open problem in [2].

5.2 Canonical Identification

Canonical identification [1] is a three-move public-key authentication protocol of a specific form. We first give canonical identification in Algorithm 2, based on the definition of Type-T signature in [2]. We add the additional checking in line 17 of Algorithm 2, which is useful for lattice-based construction. It is known that after applying the Fiat-Shamir transformation to canonical identification, we obtain a Type-T signature.

We define a new security notion of *special impersonation under key only attack* for canonical identification. It can be viewed as a combination of the special soundness and the impersonation attack: the adversary wins by outputting two valid transcripts with the same commitment.

Definition 5. *A canonical identification is secure against special impersonation under key only attack for any polynomial time adversary $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l|l} \text{VERIFY}(\mathsf{pk}, z, c) & \mathsf{param} \leftarrow \text{SETUP}(\lambda), \\ = \text{VERIFY}(\mathsf{pk}, z', c') = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow \text{KEYGEN}(), \\ \wedge\ c \neq c' \wedge c, c' \in \Delta_c & (z, c, z', c') \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{pk}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We use this new definition instead of *special soundness* together with *key recovery under key only attack* in this paper, because the standard special soundness definition [26] is not satisfied by the efficient lattice-based identification scheme used in §7. This stems from the so-called 'knowledge gap' in efficient lattice-based zero-knowledge proofs. In particular, the knowledge extractor in such schemes is *not* guaranteed to recover a secret key of a given public key, but rather recovers an 'approximate' witness of a *relaxed* relation closely related to the relation satisfied by a public-secret key pair. Therefore, to keep the generality of our results, we use the *special impersonation under key only attack*. We refer the reader to earlier works such as [19, 20, 32, 33] for further discussion about this knowledge/soundness gap issue.

We also note that for the settings where the knowledge/soundness gap issue do not arise (i.e., standard special soundness is satisfied) such as the DL-setting, 'special impersonation under key only attack' implies the standard 'key recovery under key only attack' [26] since the knowledge extractor in that case recovers a secret key sk^* with $(\mathsf{sk}^*, \mathsf{pk}) \in \text{KEYGEN}()$ given a public key pk and two accepting transcripts.

Type-T* Canonical Identification. Next, we define Type-T* canonical identification, which is a canonical identification with the following properties.

1. The function V in the verify algorithm consists of two functions V_1 and V_2 during the reconstruction of R', such that line 14 in Algorithm 2 becomes:
$$R' = V_1(z) \odot V_2(\mathsf{pk}, c),$$
where $\odot$ is a commutative group operation for the domain of R'.
2. The function V_1 is additively/multiplicatively homomorphic, i.e., $V_1(z_1) \odot V_1(z_2) = V_1(z_1 \oplus z_2)$, where $\oplus$ is the additive/multiplicative operation. The homomorphic operation should be efficiently computable.
3. Given the secret key sk corresponding to pk and c, there exists a function $\mathcal{T}$ that outputs $\hat{z} = \mathcal{T}(\mathsf{sk}, c)$ such that $V_1(\hat{z}) = V_2(\mathsf{pk}, c)$.
4. The challenge space Δ_c is a group with operation "$\otimes$". We denote the inverse operation of $\otimes$ as $\oslash$. (For example, if $\otimes$ is defined as "$+$", $\oslash$ will be "$-$".) If c_1 and c_2 are uniformly distributed in Δ_c, then $c_1 \otimes c_2$ is also uniformly distributed in Δ_c.

It is easy to see that Schnorr identification and Guillou-Quisquater identification [24] are examples of Type-T* canonical identification. There are in fact many more examples from the literature. For many identification schemes reviewed in [6], the verification function V can be split into $V_1(z)$ and $V_2(pk, c)$, such as the FFS family, FF family, and Hs family. Apart from the above schemes, the identification scheme from Katz-Wong signature [25], Chaum-Pedersen identification [15] and the Okamoto-Schnorr identification are some examples of Type-T* canonical identification. Type-T* canonical identification can also be applied to the lattice-based setting, in particular, effectively to all "Fiat-Shamir with Aborts" [32, 33]-based identification schemes (as shown in Section 7).

Schnorr identification. The SETUP algorithm outputs a cyclic group $\mathbb{G}$ of prime order p, with a generator g. For each KEYGEN execution, the algorithm picks a random $\mathsf{sk} \in \mathbb{Z}_p$ and computes $\mathsf{pk} = g^{\mathsf{sk}}$. The functions A, Z, V_1, V_2 are defined as:

$$\begin{aligned} R &= A(\mathsf{sk}; r) := g^r, \\ z &= Z(\mathsf{sk}, r, c) := r - c \cdot \mathsf{sk} \mod p, \\ R' &= V_1(z) \odot V_2(\mathsf{pk}, c) := g^z \cdot \mathsf{pk}^c. \end{aligned}$$

Note that V_1 is additively homomorphic: $V_1(z_1) \odot V_1(z_2) = g^{z_1} \cdot g^{z_2} = g^{z_1+z_2} = V_1(z_1 + z_2)$. Given the secret key sk corresponding to pk and c, it is easy to compute $\hat{z} = \mathcal{T}(\mathsf{sk}, c) := \mathsf{sk} \cdot c \bmod p$ such that $V_1(\hat{z}) = g^{\mathsf{sk}\cdot c} = \mathsf{pk}^c = V_2(\mathsf{pk}, c)$. The challenge space $\mathbb{Z}_p$ is a group under addition modulo p. Therefore, Schnorr identification is a Type-T* canonical identification.

Theorem 1. *Schnorr identification is secure against special impersonation under key only attack if the DL assumption holds.*

Proof. Suppose that $\mathcal{A}$ is an adversary breaking the special impersonation under key only attack. The algorithm $\mathcal{B}$ is given a DL problem (g, y) for a cyclic group $\mathbb{G}$ of prime order p. $\mathcal{B}$ gives $\mathsf{param} = (\mathbb{G}, p, g)$ and $\mathsf{pk} = y$ to $\mathcal{A}$.

$\mathcal{A}$ returns (c, z, c', z') where $c \neq c'$. Then we have:

$$g^z \cdot \mathsf{pk}^c = g^{z'} \cdot \mathsf{pk}^{c'}.$$

Therefore $\mathcal{B}$ can extract the secret key $\mathsf{sk} = \frac{z-z'}{(c'-c)}$ as the solution to the DL problem. □

Guillou-Quisquater (GQ) identification [24]. The Setup algorithm outputs a pair (N, e), where $N = pq$, p and q are large prime numbers, e is a prime number less than $N/4$ and $\gcd(e, \phi(N)) = 1$. For each KeyGen execution, the algorithm picks a random $\mathsf{sk} \in \mathbb{Z}_N$ and calculates $\mathsf{pk} = \mathsf{sk}^e$. The functions A, Z, V_1, V_2 are defined as:

$$\begin{aligned} R &= A(\mathsf{sk}; r) := r^e, \\ z &= Z(\mathsf{sk}, r, c) := \mathsf{sk}^c \cdot r \mod N, \\ R' &= V_1(z) \odot V_2(\mathsf{pk}, c) := z^e \cdot \mathsf{pk}^{-c} \mod N. \end{aligned}$$

Note that V_1 is multiplicatively homomorphic: $V_1(z_1) \odot V_1(z_2) = z_1^e \cdot z_2^e = (z_1 z_2)^e = V_1(z_1 \cdot z_2)$. Given the secret key sk corresponding to pk and c, it is easy to compute $\hat{z} = \mathcal{T}(\mathsf{sk}, c) := \mathsf{sk}^{-c} \bmod N$ such that $V_1(\hat{z}) = \hat{z}^e = \mathsf{sk}^{-ce} = \mathsf{pk}^{-c} = V_2(\mathsf{pk}, c)$. The challenge space of GQ identification is $\mathbb{Z}_e$ and it is a group under addition. Therefore, GQ identification is a Type-T* canonical identification.

Theorem 2. *GQ identification is secure against special impersonation under key only attack if the RSA assumption holds.*

Proof. Suppose that $\mathcal{A}$ is an adversary breaking the special impersonation under key only attack. The algorithm $\mathcal{B}$ is given a RSA problem (N, e, y). $\mathcal{B}$ gives $\mathsf{param} = (N, e)$ and $\mathsf{pk} = y$ to $\mathcal{A}$.

$\mathcal{A}$ returns (c, z, c', z'), where $c \neq c'$, we have $z^e \cdot \mathsf{pk}^{-c} = z'^e \cdot \mathsf{pk}^{-c'}$. Then:

$$(z/z')^e = \mathsf{pk}^{(c-c')}$$

Since e is a prime and $c, c' \in \mathbb{Z}_e$, $\mathcal{B}$ can compute integers A and B such that:

$$A \cdot e + B \cdot (c - c') = \gcd(e, (c - c')) = 1,$$

by the Euclidean algorithm. Hence we have:

$$(z/z')^{Be} = \mathsf{pk}^{1-Ae}.$$

Therefore we can extract the secret key $\mathsf{sk} = (z/z')^B \mathsf{pk}^A$ as the solution to the RSA problem. □

5.3 Our Construction: DualRing

We denote a Type-T* canonical identification scheme by T*. We use the symbol $\bigodot$ and $\bigotimes$ to represent consecutive $\odot$ and $\otimes$ operations, respectively:

$$\bigodot_{i=1}^{n} a_i := a_1 \odot a_2 \odot \ldots \odot a_{n-1} \odot a_n, \quad \bigotimes_{i=1}^{n} b_i := b_1 \otimes b_2 \otimes \ldots \otimes b_{n-1} \otimes b_n.$$

Algorithm 3: DualRing

```
Procedure $\textsc{Setup}(\lambda)$:
    define $H : \{0,1\}^* \to \Delta_c$;
    return $\mathsf{param} \leftarrow \mathrm{T}^*.\textsc{Setup}(\lambda)$;
Procedure $\textsc{Sign}(\mathsf{param}, m, \mathbf{pk} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{sk}_j)$:
    $r \leftarrow_s \Delta_r, c_i \leftarrow_s \Delta_c$ for all $i \neq j$;
    $R = A(\mathsf{sk}_j; r) \odot \bigodot_{i \neq j} V_2(\mathsf{pk}_i, c_i)$;
    $c = H(m, \mathbf{pk}, R)$;
    $c_j = c \oslash \bigotimes_{i \neq j} c_i$;
    $z = Z(\mathsf{sk}_j, r, c_j)$;
    return $\sigma = (c_1, \ldots, c_n, z)$;
Procedure $\textsc{KeyGen}(\mathsf{param})$:
    return $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{T}^*.\textsc{KeyGen}(\mathsf{param})$;
Procedure $\textsc{Verify}(\mathsf{param}, m, \mathbf{pk} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \sigma)$:
    parse $\sigma = (c_1, \ldots, c_n, z)$;
    $R = V_1(z) \odot \bigodot_{i=1}^{n} V_2(\mathsf{pk}_i, c_i)$;
    $c = \bigotimes_{i=1}^{n} c_i$;
    if $c \neq H(m, \mathbf{pk}, R)$ then
        return 0;
    auxiliary checking with $(R, c, z)$;
    return 1;
```

DualRing is shown in Algorithm 3. The high level idea is that we use V_2 to add the decoy public keys pk_i and their corresponding challenge values c_i to the commitment R first. After getting the real challenge value c, the signer with index j computes $c_j = c \oslash \bigotimes_{i \neq j} c_i \in \Delta_c$. The signer computes z according to the canonical identification scheme. To verify, the commitment R is reconstructed from all public keys and their corresponding challenge values. The value $\bigotimes_{\forall i} c_i$ should be equal to the real challenge value c.

Theorem 3. *DualRing is unforgeable w.r.t. insider corruption in the random oracle model if T^* is secure against special impersonation under key only attack and $|\Delta_c| > q_s(q_h + q_s - 1)$, where q_s and q_h are the number of queries to the signing oracle and the H oracle respectively.*

Proof. Denote $\mathcal{A}$ as a PPT adversary breaking the unforgeability w.r.t. insider corruption of DualRing. We build an algorithm $\mathcal{B}$ to break the special impersonation under key only attack of T*. Suppose the algorithm $\mathcal{B}$ is given a system parameters param and a public key pk^* from its challenger $\mathcal{C}$.

Setup. $\mathcal{B}$ picks a random index $i^* \in [1, q_k]$. $\mathcal{B}$ runs $(\hat{\mathsf{pk}}_i, \hat{\mathsf{sk}}_i) \leftarrow \mathsf{KeyGen}()$ for $i \in [1, q_k], i \neq i^*$. $\mathcal{B}$ sets $\hat{\mathsf{pk}}_{i^*} = \mathsf{pk}^*$. $\mathcal{B}$ gives param and $S := \{\hat{\mathsf{pk}}_i\}_{i=1}^{q_k}$ to $\mathcal{A}$.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

- H: $\mathcal{B}$ simulates H as a random oracle.
- $\mathcal{CO}$: On input i, $\mathcal{B}$ returns $\hat{\mathsf{sk}}_i$ (If $i = i^*$, $\mathcal{B}$ declares failure and exits.).
- $\mathcal{SO}$: On input a message M, a set of public key $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and the signer index j, it outputs $\perp$ if $\hat{\mathsf{pk}}_j \notin \mathbf{pk}$. If $j \neq i^*$, then $\mathcal{B}$ returns $\sigma \leftarrow \textsc{Sign}(\mathsf{param}, M, \mathbf{pk}, \hat{\mathsf{sk}}_j)$.
  Otherwise, $\mathcal{B}$ picks random $c_1, \ldots, c_n \in \Delta_c$ and z from the domain of response Δ_z according to the distribution of the output of $Z(\cdot)$. $\mathcal{B}$ computes $R = V_1(z) \odot \bigodot_{i=1}^{n} V_2(\mathsf{pk}_i, c_i)$. $\mathcal{B}$ sets $H(M, \mathbf{pk}, R) = \bigotimes_{i=1}^{n} c_i$ in the random oracle. If such value has been set in the random oracle, $\mathcal{B}$ declares failure and exits. $\mathcal{B}$ returns $\sigma = (c_1, \ldots, c_n, z)$.

Challenge. $\mathcal{A}$ returns a forgery $(M^*, \{\hat{\mathsf{pk}}_{i_j}\}_{j=1}^{n}, \sigma^* = (c_1^*, \ldots, c_{n^*}^*, z^*))$. If $\mathsf{pk}^* \neq \hat{\mathsf{pk}}_{i_j}$ for all $j \in [1, n]$, $\mathcal{B}$ declares failure and exits. Otherwise, we denote j^* as the index such that $\mathsf{pk}^* = \hat{\mathsf{pk}}_{i_{j^*}}$. Denote $\mathbf{pk}^* = \{\hat{\mathsf{pk}}_{i_j}\}_{j=1}^{n}$ and compute R^* as in the $\textsc{Verify}$ algorithm. $\mathcal{B}$ rewinds to the point that $(M^*, \mathbf{pk}^*, R^*)$ is queried to H and returns a different c' instead. $\mathcal{A}$ returns another signature $\sigma' = (c_1', \ldots, c_n', z')$. Since both σ^* and σ' are valid signatures, We have:

$$R^* = V_1(z^*) \odot \bigodot_{j=1}^{n} V_2(\hat{\mathsf{pk}}_{i_j}, c_j^*) = V_1(z') \odot \bigodot_{j=1}^{n} V_2(\hat{\mathsf{pk}}_{i_j}, c_j').$$

Note that it is impossible to have $c_j^* = c_j'$ for all $j \in [1,n]$ (since $\bigotimes_{j=1}^n c_j^* \neq \bigotimes_{j=1}^n c_j'$). If $c_{j^*}^* = c_{j^*}'$, $\mathcal{B}$ declares failure and exits. With probability at least $1/n$, we have $c_{j^*}^* \neq c_{j^*}'$. Observe that:

$$\begin{aligned}&V_1(z^*) \odot \bigodot_{j=1}^{n} V_2(\hat{\mathsf{pk}}_{i_j}, c_j^*)\\ =&V_1(z^* \oplus \hat{z}_1^* \oplus \ldots \oplus \hat{z}_{j^*-1}^* \oplus \hat{z}_{j^*+1}^* \oplus \ldots \oplus \hat{z}_n^*) \odot V_2(\mathsf{pk}^*, c_{j^*}^*)\\ =&V_1(\tilde{z^*}) \odot V_2(\mathsf{pk}^*, c_{j^*}^*),\end{aligned}$$

where $\hat{z}_i^* = \mathcal{T}(\hat{\mathsf{sk}}_i, c_i^*)$ for $i \in [1,n] \setminus j^*$ and $\tilde{z^*} = z^* \oplus \hat{z}_1^* \oplus \ldots \oplus \hat{z}_{j^*-1}^* \oplus \hat{z}_{j^*+1}^* \oplus \ldots \oplus \hat{z}_n^*$. Similarly we have $V_1(z') \odot \bigodot_{j=1}^n V_2(\hat{\mathsf{pk}}_{i_j}, c_j') = V_1(\tilde{z'}) \odot V_2(\mathsf{pk}^*, c_{j^*}')$ for some $\tilde{z'}$. Then $\mathcal{B}$ can return $(c_{j^*}^*, \tilde{z^*}, c_{j^*}', \tilde{z'})$ to its challenger $\mathcal{C}$.

Probability Analysis. We analyse the probability of success (i.e., not aborting) in the above simulation. For q_c queries to the $\mathcal{CO}$, the probability of success in the first query is $(1 - \frac{1}{q_k})$. The probability of success in the second query is at least $(1 - \frac{1}{q_k - 1})$. The probability of success after q_c queries is at least $(1-\frac{1}{q_k})(1-\frac{1}{q_k-1})\cdots(1-\frac{1}{q_k-q_c+1}) = \frac{q_k-q_c}{q_k} = 1 - \frac{q_c}{q_k}$. (It is implied by the security model that $q_k > q_c + n$.)

For q_s queries to the $\mathcal{SO}$, the probability of success in the first query is at least $(1 - \frac{q_h}{|\Delta_c|})$, where q_h is the number queries to the H oracle. The probability of success after q_s queries to $\mathcal{SO}$ is at least

$$(1-\frac{q_h}{|\Delta_c|})(1-\frac{q_h+1}{|\Delta_c|})\cdots(1-\frac{q_h+q_s-1}{|\Delta_c|}) \geq (1-\frac{q_h+q_s-1}{|\Delta_c|})^{q_s} \geq 1-\frac{q_s(q_h+q_s-1)}{|\Delta_c|}.$$

Here we assume that $|\Delta_c| > q_s(q_h + q_s - 1)$.

The probability of $\mathsf{pk}^* \neq \hat{\mathsf{pk}}_{i_j}$ in the challenge phase is $(1 - \frac{1}{q_k - q_c})(1 - \frac{1}{q_k-q_c-1})\cdots(1 - \frac{1}{q_k-q_c-n+1}) = \frac{q_k-q_c-n}{q_k-q_c}$. If the probability of forgery by $\mathcal{A}$ is ϵ, then the probability of $\mathcal{B}$ does not return failure before rewinding is

$$\begin{aligned}\epsilon_b &:= \epsilon(1-\frac{q_c}{q_k})(1-\frac{q_s(q_h+q_s-1)}{|\Delta_c|})(1-\frac{q_k-q_c-n}{q_k-q_c})\\ &= \epsilon(1-\frac{q_s(q_h+q_s-1)}{|\Delta_c|})(\frac{n}{q_k}).\end{aligned}$$

By the generalized forking lemma [4], the probability of a successful rewinding is at least $\frac{\epsilon_b}{8}$ if $|\Delta_c| > 8q_h/\epsilon_b$ (it runs in time $\tau \cdot 8q_n/\epsilon_b \cdot \ln(8n/\epsilon_b)$ if $\mathcal{A}$ runs in time τ). Finally we have $c_{j^*}^* \neq c_{j^*}'$ with probability at least $1/n$. As a result, the probability ϵ' of $\mathcal{B}$ breaking the special impersonation is:

$$\epsilon' \geq (\frac{\epsilon n}{8q_k})(1-\frac{q_s(q_h+q_s-1)}{|\Delta_c|}).$$

if $|\Delta_c| > q_s(q_h + q_s - 1)$ and $|\Delta_c| > 8q_h/\epsilon_b$[8]. We can further simplify the probability ϵ' if we take $|\Delta_c| \geq 2q_s(q_h + q_s - 1)$. Then if $|\Delta_c| > \frac{16q_hq_k}{\epsilon n}$, we have $\epsilon' \geq \frac{\epsilon n}{16q_k}$. □

Theorem 4. *DualRing is anonymous in the random oracle model, if $|\Delta_c| > q_s(q_h+q_s-1)$, where q_s and q_h are the number of queries to the signing oracle and the H oracle respectively.*

Proof. We show how to build an algorithm $\mathcal{B}$ providing perfect anonymity in the random oracle model.

Setup. $\mathcal{B}$ runs $\mathsf{param} \leftarrow \textsc{Setup}(\lambda)$. $\mathcal{B}$ runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param}; \omega_i)$ for $i \in [1, q_k]$ with randomness ω_i. $\mathcal{B}$ gives param and $S := \{\mathsf{pk}_i\}_{i=1}^{q_k}$ to $\mathcal{A}_1$.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

[8] The condition $|\Delta_c| > 8q_h/\epsilon_b$ is not needed if we use the forking lemma in [7] with a looser security bound.

– $\mathcal{SO}$: On input a message m, a set of public key $\mathbf{pk}$ with the signer index j, $\mathcal{B}$ returns $\sigma \leftarrow$ $\textsc{Sign}(\mathsf{param}, m, \mathbf{pk}, \mathsf{sk}_j)$.
– H: $\mathcal{B}$ simulates H as a random oracle.

Challenge. $\mathcal{A}_1$ gives $\mathcal{B}$ a message m and a vector of public keys $\mathbf{pk}$ and two indices i_0, i_1. $\mathcal{B}$ picks random $c_1, \ldots, c_n \in \Delta_c$ and picks z from the domain of response Δ_z according to the distribution of the output of $Z(\cdot)$. $\mathcal{B}$ computes $R = V_1(z) \odot \bigodot_{i=1}^{n} V_2(\mathsf{pk}_i, c_i)$. $\mathcal{B}$ sets $H(m, \mathbf{pk}, R) = \bigotimes_{i=1}^{n} c_i$ in the random oracle. By Property 4, the distribution is correct. If the hash value is already set by the H oracle, $\mathcal{B}$ declares failure and exits. $\mathcal{B}$ returns $\sigma = (c_1, \ldots, c_n, z)$ and $\{\omega_i\}_{i=1}^{q_k}$ to $\mathcal{A}_2$. $\mathcal{B}$ picks a random bit b.

Output. Finally, $\mathcal{A}_2$ outputs a bit b'. Observe that b is not used in the generation of σ. Therefore, $\mathcal{A}_2$ can only win with probability 1/2.

Probability Analysis. We analyse the probability of success (i.e., not aborting) in the above simulation. For q_h queries to the H oracle and q_s queries to the $\mathcal{SO}$, the probability of success in the first query is at least $(1 - \frac{q_h}{|\Delta_c|})$. The probability of success after q_s queries to $\mathcal{SO}$ is at least

$$(1 - \frac{q_h}{|\Delta_c|})(1 - \frac{q_h + 1}{|\Delta_c|}) \cdots (1 - \frac{q_h + q_s - 1}{|\Delta_c|}) \geq (1 - \frac{q_h + q_s - 1}{|\Delta_c|})^{q_s} \geq 1 - \frac{q_s(q_h + q_s - 1)}{|\Delta_c|}.$$

Here we assume that $|\Delta_c| > q_s(q_h + q_s - 1)$. If $\mathcal{B}$ does not abort, then no PPT adversary can win with non-negligible probability over half. □

Difference with AOS Ring Signature. Our ring signature is a bit different from the AOS ring signature. The AOS ring signature allows a mixture of different types of public keys, such as keys from the Schnorr signature and the RSA signature. The security proof for the generic construction of the AOS ring signature was not formally given in [2]. On the other hand, our scheme allows different types of public keys from different Type-T* canonical identification schemes, with the restriction that these canonical identification schemes should use the same V_1 function[9] (Otherwise, we do not know which V_1 function to use in the $\textsc{Verify}$ algorithm). The security proof for our generic construction holds for different Type-T* canonical identification schemes satisfying the requirement above.

The AOS ring signature is generated sequentially by forming a "ring" of c_i in a loop and calculating z_i for n times. On the other hand, our signature is generated by forming a "R-ring" in one-shot during the commit phase, forming a "C-ring" in one-shot after getting the challenge c and calculating z for one time only. Therefore, our scheme is more efficient than the AOS ring signature.

Finally, our dual ring technique cannot be applied to the Type-H signature in [2]. Recall that for our Type-T* DualRing, we require the use of $V_2(\mathsf{pk}_i, c_i)$ (for all non-signer indices) to generate R. For Type-H, pk_i is tied with z by the one-way function $F(z, \mathsf{pk}_i)$. Hence, we cannot separate z and pk_i into V_1 and V_2 to form the R-ring similarly.

Difference with CDS OR-proofs. The C-Ring in DualRing is similar to the use of secret sharing in CDS 1-out-of-n OR proof [16]. Our construction of R-Ring leads to a single R and hence a single z in the signature. On the contrary, [16] does not have the formation of R-Ring and still has n commitments R_i's. It results in n responses z_i's. So, the ring signature constructed by [16] consists of (c_i, z_i) for $i \in [n]$. There is no trivial way to combine all z_i's, because each z_i is only related to R_i and c_i, and not to other z_j's. Hence, [16] does not (easily) achieve an $O(\log n)$ size ring signature in the DL-based setting.

[9] which implicitly implies all users should use the same set of security parameters including the same group and generator for their sk and pk.

5.4 Update over the Conference Version

Here, we describe an improvement over our DualRing construction provided in the conference version of this work [38]. The algebraic properties needed from the challenge space (Property 4 of Type-T* canonical identification) can actually be removed using the following simple "trick". Assume that we want the challenge space to have 2^κ elements for some parameter κ.[10] Then, we define a bijective map $\mathcal{G} : \mathbb{Z}_2^\kappa \to \Delta_c$. Let us call a pre-image of this map as a "pre-challenge" and its output as a "challenge" as before. Then, we define the random oracle H to output values in $\mathbb{Z}_2^\kappa$. Overall, in signing, the signer samples pre-challenge $\hat{c}_i$'s (for $i \neq j$) as κ-bit random strings, gets a random κ-bit real pre-challenge $\hat{c}$ as the output of the random oracle H and computes $\hat{c}_j = \hat{c} \oplus \bigoplus_{i \neq j} \hat{c}_i$ where $\oplus$ and $\bigoplus$ now simply denote the XOR operation. From $\hat{c}_i$'s, it is straightforward to compute the challenges $c_i = \mathcal{G}(\hat{c}_i)$ needed for the functions V_2 and Z. The signer in the end outputs pre-challenges in the signature, i.e., $\sigma = (\hat{c}_1, \ldots, \hat{c}_n, z)$. Similarly, verification is modified to compute the real pre-challenge $\hat{c} = \bigoplus_{i=1}^n \hat{c}_i$, and check that it matches the random oracle output. The above changes do not affect the security proofs and they still follow with minor modifications.

This technique helps to make our DualRing construction more generic by removing the requirement for Property 4 of Type-T* canonical identification. Although it is not really helpful for our DL-based instantiation DualRing-EC, it provides more flexibility in choosing the parameters for our lattice-based instantiation DualRing-LB (see Appendix D).

6 DualRing-EC: Our Succinct DL-based Ring Signature

We give a new *sum argument of knowledge* which is useful to reduce the signature size of DualRing from linear to logarithmic in the DL-based setting. The group operation $\otimes$ of the challenge space is modular addition. This is the first combination of the classical ring structure with the argument of knowledge.

Notations and Assumptions. For a security parameter λ, we use $\mathbb{G}$ to represent a cyclic group of prime order p. We use $[n]$ to denote the numbers $1, 2, ..., n$.

We use the following notations for vectors for our DL-based construction: $\mathbf{a}_{[:l]}$ and $\mathbf{a}_{[l:]}$ represent $(a_1, ..., a_l)$ and $(a_{l+1}, ..., a_n)$. $\mathbf{a} \circ \mathbf{b}$ is the Hadamard product $(a_1 b_1, a_2 b_2, ..., a_n b_n)$. $\langle \mathbf{a}, \mathbf{b} \rangle$ is the inner product $\sum_{i=1}^n a_i b_i$. $\mathbf{a}^b$, $\mathbf{a} + b$ and $\mathbf{a}^{\mathbf{b}}$ represent $(a_1^b, a_2^b, ..., a_n^b)$, $(a_1 + b, a_2 + b, ..., a_n + b)$ and $\prod_{i=1}^n a_i^{b_i}$ respectively. $\sum \mathbf{a}$ and $\prod \mathbf{a}$ denotes $\sum_{i=1}^n a_i$ and $\prod_{i=1}^n a_i$.

Definition 6 (Discrete Logarithm Assumption). *For all PPT adversaries $\mathcal{A}$ such that*

$$\Pr\left[y = g^a | g, y \leftarrow_s \mathbb{G}, a \leftarrow \mathcal{A}(\mathbb{G}, g, y)\right] \leq \mathsf{negl}(\lambda).$$

6.1 Sum Arguments of Knowledge

The sum argument of knowledge is a variant of inner product argument in [14]. The inner product argument is an efficient proof system for the following relation:

$$\left\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\right\}$$

in which a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ that c is the inner product of two committed vectors $\mathbf{a}, \mathbf{b}$. Bootle *et al.* [12] presented an efficient zero-knowledge proof for inner product argument, with communication complexity of $6 \log_2(n)$ (n is the dimension of vectors). Based on their works, Bünz *et al.* proposed Bulletproofs [14] to reduce the communication complexity to $2 \log_2(n)$. They achieve $O(\log n)$ complexity by running a recursive Pf algorithm, such that in each round two vectors $\mathbf{a}, \mathbf{b}$ of size n are committed into two commitments (L, R), and two vector of proofs $\mathbf{a}', \mathbf{b}'$ of size $n/2$ are computed for challenge x, where $L^{x^2} P R^{x^{-2}}$ is equal to the commitment of $\mathbf{a}', \mathbf{b}'$ and $\langle \mathbf{a}', \mathbf{b}' \rangle$. In the next round, run the Pf algorithm with input vectors $\mathbf{a}', \mathbf{b}'$ and the recursion ends when $n = 1$.

[10] We could also assume that $|\Delta_c| = m^\kappa$ for $m, \kappa \in \mathbb{Z}^+$, but choose $m = 2$ for simplicity.

From Inner Product Argument to Sum Argument. To construct our logarithmic size ring signature, we propose a new argument of knowledge named Sum Argument. First we give the relation:

$$\Big\{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \wedge c = \sum \mathbf{a}\Big\} \tag{2}$$

In a sum argument, a prover convinces a verifier that he/she has the knowledge of a vector of scalars $\mathbf{a}$, such that $P = \mathbf{g}^{\mathbf{a}}$ and $c = \sum \mathbf{a}$. Our sum argument looks like an inner product argument, where a vector of generators and a computation of multi-exponentiation is used. Although an inner product argument can be converted into a sum argument by setting the vector $\mathbf{b}$ to $\mathbf{1}^n$, this yields a less efficient protocol than ours.

Assume that the system parameter param includes a generator $u \in \mathbb{G}$ in group $\mathbb{G}$ with order p and two hash functions $H_Z, H'_Z : \{0,1\}^* \to \mathbb{Z}_p$. A Non-interactive Sum Argument (NISA) consists of a Proof algorithm which takes $(\mathsf{param}, \mathbf{g}, P, c, \mathbf{a})$ and outputs a proof π; and a Verify algorithm which takes$(\mathsf{param}, \mathbf{g}, P, c, \pi)$ and outputs a bit 1/0. Our NISA is given in Algorithm 4. Observe that for the k-th recursion in Pf, the value of $\mathbf{b}$ is $\prod_{i=1}^{k}(x_i + x_i^{-1})\mathbf{1}^{\frac{n}{2^k}}$, where x_i is the i-th output of H_Z. This $\mathbf{b}$ is known to the verifier and hence we do not need a vector of generators $\mathbf{h}$ to commit $\mathbf{b}$ in L, R as in [12]. As a result, we can set $\mathbf{h}$ as $\mathbf{1}^n$ and can save almost half of the exponentiation during the recursion. In addition, the computation of P is also not needed by the prover.

Theorem 5. *Our sum argument has statistical witness-extended emulation for non-trivial discrete logarithm relation among* $\mathbf{g}, u$ *or a valid witness* $\mathbf{a}$.

We defer its security proof to the Appendix B.

Compared with [14], our protocol is simpler. In each iteration, we compute $(4n'+2)$ exponentiations to generate a proof, then compute a multi-exponentiation of size $(1+n+2\log_2(n))$ to verify. For an inner product argument [14], the corresponding computations are $(8n'+8)$ exponentiations and a multi-exponentiation of size $(1+2n+2\log_2(n))$, respectively. The proof sizes are similar; however we omit almost half of exponentiations.

6.2 Logarithmic Size DL-based Ring Signature

We give the full construction of compact DL-based ring signature, by combining DualRing with the sum argument of knowledge and Schnorr identification. Then, we compare the efficiency with the existing ring signature schemes.

Matching Sum Argument with Ring Signature. Notice that the sum argument proves the relation for some $a_i \in \mathbb{Z}_p$, given $g_i, P \in \mathbb{G}$ and $c \in \mathbb{Z}_p$:

$$P = \prod_{i=1}^{n} g_i^{a_i} \quad \wedge \quad c = \sum_{i=1}^{n} a_i.$$

On the other hand, the verification of our generic ring signature includes:

$$R \odot (V_1(z))^{-1} = \bigodot_{i=1}^{n} V_2(\mathsf{pk}_i, c_i) \quad \wedge \quad H(m, \mathsf{pk}, R) = \bigotimes_{i=1}^{n} c_i.$$

Interestingly, the two examples (DL- and RSA-based) of the $\mathsf{Type\text{-}T^*}$ canonical identification have $V_2(\mathsf{pk}_i, c_i) = \mathsf{pk}_i^{c_i}$. Therefore, we can use the sum argument for the relation:

$$R \cdot (V_1(z))^{-1} = \prod_{i=1}^{n} \mathsf{pk}_i^{c_i} \quad \wedge \quad H(m, \mathsf{pk}, R) = \sum_{i=1}^{n} c_i.$$

As a result, we can give a logarithmic size ring signature from $\mathsf{Type\text{-}T^*}$ canonical identification scheme with matching non-interactive sum argument.

DualRing-EC Construction. Our DL-based construction DualRing-EC is shown in Algorithm 5, by using DualRing and NISA for Relation 2.

Algorithm 4: NISA

```
1  Procedure NISA.Proof($\{\mathsf{param}, \mathbf{g}, P, c\}, \mathbf{a}$):
2      Run protocol Pf on input $(\mathbf{g}, u^{H'_Z(P,u,c)}, \mathbf{a}, \mathbf{1}^n)$;
3  Procedure Pf($\mathbf{g}, \hat{u}, \mathbf{a}, \mathbf{b}$):
       // L, R are initially empty, but maintain its memory throughout the recurrsion.
          $n$ is the length of vector a and b.
4      if $n = 1$ then
5          Output $\pi = (\mathbf{L}, \mathbf{R}, a, b)$.
6      else
7          Compute $n' = \frac{n}{2}$, $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$, $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$;
8          $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \hat{u}^{c_L} \in \mathbb{G}$ and $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \hat{u}^{c_R} \in \mathbb{G}$;
9          Add $L$ to $\mathbf{L}$ and $R$ to $\mathbf{R}$ and compute $x = H_Z(L, R)$;
10         Compute $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$, $\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$ and
           $\mathbf{b}' = x^{-1} \cdot \mathbf{b}_{[:n']} + x \cdot \mathbf{b}_{[n':]} \in \mathbb{Z}_p^{n'}$;
11         Run protocol Pf on input $(\mathbf{g}', \hat{u}, \mathbf{a}', \mathbf{b}')$;
12 Procedure NISA.Verify($\mathsf{param}, \mathbf{g}, P, c, \pi = (\mathbf{L}, \mathbf{R}, a, b)$):
13     $P' = P \cdot u^{c \cdot H'_Z(P,u,c)}$;
14     Compute for all $j = 1, ..., \log_2 n$: $x_j = H_Z(L_j, R_j)$;
15     Compute for all $i = 1, ..., n$: $y_i = \prod_{j \in [\log_2 n]} x_j^{f(i,j)}, f(i,j) = \begin{cases} 1 & \text{if } (i-1)\text{'s } j\text{-th bit is } 1 \\ -1 & \text{otherwise} \end{cases}$;
16     Set $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{x} = (x_1, \ldots, x_{\log_2 n})$ ;
17     if $\mathbf{L}^{\mathbf{x}^2} P' \mathbf{R}^{\mathbf{x}^{-2}} = \mathbf{g}^{a \cdot \mathbf{y}} u^{ab \cdot H'_Z(P,u,c)}$ then
18         Output 1
19     Output 0
```

Theorem 6. *DualRing-EC is unforgeable w.r.t. insider corruption if* DualRing *is unforgeable w.r.t. insider corruption and the NISA has statistical witness-extended emulation.*

Proof. Suppose that $\mathcal{A}$ is an adversary breaking the unforgeability w.r.t. insider corruption of DualRing-EC. Then, we can construct an algorithm $\mathcal{B}$ breaking the unforgeability of DualRing. $\mathcal{B}$ is given the system parameter param' and a set of public keys S from the challenger of DualRing. $\mathcal{B}$ picks a random generator $u \in \mathbb{G}$ and returns $\mathsf{param} = (\mathsf{param}', u)$ to $\mathcal{A}$.

When $\mathcal{A}$ asks for a signing oracle query, $\mathcal{B}$ asks the signing oracle of DualRing and obtains $\sigma' = (c_1, \ldots, c_n, z)$. $\mathcal{B}$ computes R by running DualRing.Verify on σ'. $\mathcal{B}$ computes $c = c_1 + \cdots + c_n$ and $P = R \odot (V_1(z))^{-1}$. $\mathcal{B}$ runs the NISA.Proof and obtains π. $\mathcal{B}$ returns (c, z, R, π).

In the challenge phase, $\mathcal{A}$ returns a signature $\sigma^* = (c^*, z^*, R^*, \pi^*)$ with respect to a message M^* and $\{\mathsf{pk}_i^*\}_{i=1}^n$. By the statistical witness-extended emulation of NISA, $\mathcal{B}$ can run an extractor $\mathcal{E}$ to obtain $(c_1^*, \ldots, c_n^*)$, where $P^* = R^* \odot (V_1(z^*))^{-1} = \bigodot_{i=1}^n V_2(\mathsf{pk}_i^*, c_i^*)$. Then $\mathcal{B}$ returns the signature $\sigma' = (c_1^*, \ldots, c_n^*, z^*)$, the message M^* and $\{\mathsf{pk}_i^*\}_{i=1}^n$ to the challenger of DualRing. □

Theorem 7. *DualRing-EC is anonymous if* DualRing *is anonymous.*

Proof. Suppose that $\mathcal{A}$ is an adversary breaking the anonymity of DualRing-EC. Then, we can construct an algorithm $\mathcal{B}$ breaking the anonymity of DualRing. $\mathcal{B}$ is given param' and the set S from its challenger. $\mathcal{B}$ picks a random generator $u \in \mathbb{G}$ and gives $\mathsf{param} = (\mathsf{param}', u)$ to $\mathcal{A}$.

When $\mathcal{A}$ asks for a signing oracle query, $\mathcal{B}$ simulates it as in the proof of unforgeability. In the challenge phase, $\mathcal{A}$ gives $M^*, \vec{\mathsf{pk}}^*, i_0, i_1)$ to $\mathcal{B}$ and $\mathcal{B}$ forwards it to its challenger. $\mathcal{B}$ receives $((c_1^*, \ldots, c_n^*, z^*), \{\omega_i\}_{i=1}^{q_k})$. $\mathcal{B}$ computes σ^* by line 7-9 of the Sign algorithm and returns $(\sigma^*, \{\omega_i\}_{i=1}^{q_k})$ to $\mathcal{A}$.

Finally $\mathcal{A}$ returns a bit b' and $\mathcal{B}$ sends b' to its challenger to break the anonymity of DualRing. □

Algorithm 5: DualRing-EC

```
1  Procedure Setup($\lambda$):
2      $\mathsf{param}' \leftarrow$ DualRing.Setup($\lambda$);
3      pick a generator $u \leftarrow_s \mathbb{G}$;
4      return $\mathsf{param} = (\mathsf{param}', u)$;
5  Procedure Sign($\mathsf{param}, m, \mathbf{pk}, sk_j$):
6      $(c_1, \ldots, c_n, z) \leftarrow$ DualRing.Sign
         $(\mathsf{param}, m, \mathbf{pk}, sk_j)$;
       // $(c, R)$ is computed in DualRing.Sign
7      $\mathbf{a} \leftarrow (c_1, \ldots, c_n)$;
8      $P = R \odot (V_1(z))^{-1}$;
9      $\pi \leftarrow$ NISA.Proof($\{\mathsf{param}, \mathbf{pk}, u, P, c\}, \mathbf{a}$);
10     return $\sigma = (z, R, \pi)$;
11 Procedure KeyGen($\mathsf{param}$):
12     return $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ DualRing.KeyGen($\mathsf{param}'$);
13 Procedure Verify($\mathsf{param}, m, \mathbf{pk}, \sigma$):
14     parse $\sigma = (z, R, \pi)$;
15     $c = H_Z(m, \mathbf{pk}, R)$;
16     $P = R \odot (V_1(z))^{-1}$;
17     if $0 \leftarrow$ NISA.Verify($\mathsf{param}, \mathbf{pk}, u, P, c$) then
18         return 0;
19     return 1;
```

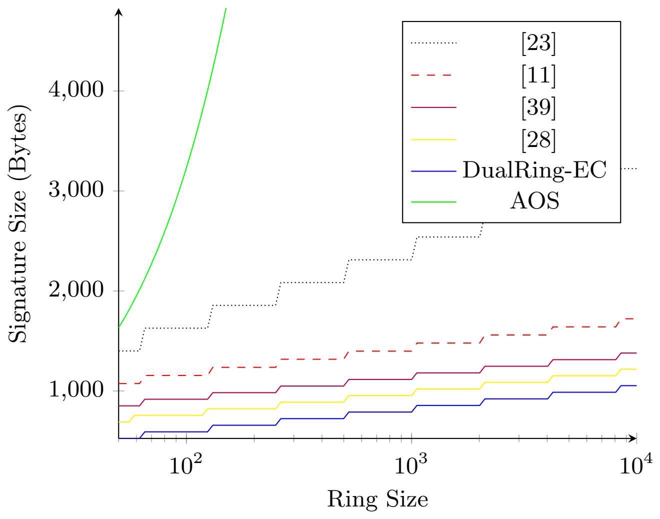


Fig. 3: The signature size of ring signature schemes for n public keys, when implemented on elliptic curve with $\lambda = 128$.

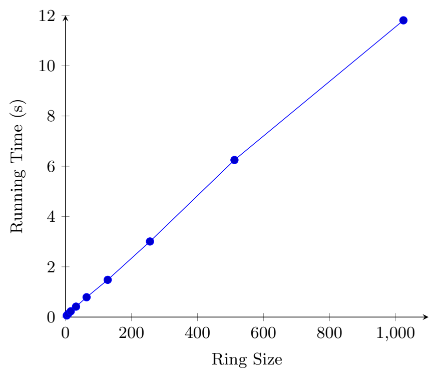
6.3 Efficiency Analysis

Signature Size. We compare our DL-based instantiation for n public keys with other $O(\log n)$-size DL-based ring signatures without trusted setup in Table 1. Most accumulator-based $O(1)$-size ring signatures require trusted setup. The lattice-based logarithmic ring signatures [19, 20, 29] are still at least 100 times longer than DL-based construction. Our ring signature is 789/921 bytes for the ring size = 1024/4096 with $\lambda = 128$. We can see that DualRing-EC (Algorithm 5 with Schnorr Identification) is the shortest ring signature without trusted setup. Fig. 3 shows the concrete signature size when an element in $\mathbb{Z}_p$ is represented by 32 bytes and an element in $\mathbb{G}$ is represented by 33 bytes. Note that the signature size for a ring with size $[\log(n-1)+1, \log n]$ is the same. Therefore, the signature size increases for ring size 1025, 2049, 4097, etc.
Computational Efficiency. We implement our DualRing-EC in Python, using the P256 curve in the fastecdsa library. It is tested on a computer with Intel Core i5 2.3GHz, 8GB RAM with MacOS 10. The running time is shown in Fig. 4.

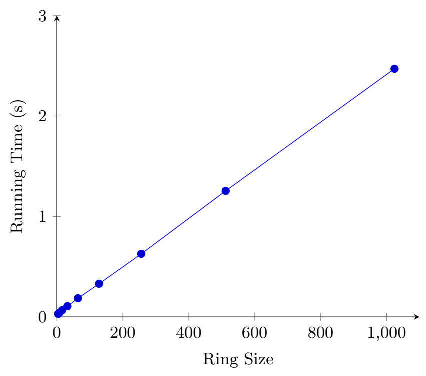
We compare the asymptotic running time of our scheme with [11, 23] [11]. The running time of the signer for both [23] and [11] are both dominated by $O(n \log n)$ exponentiations. On the

[11] For simplicity, we compare the schemes by assuming that a multi-exponentiation of size ℓ is the same as ℓ exponentiation in $\mathbb{G}$.

other hand, the signer's running time for DualRing-EC is $O(n)$ exponentiations only. Comparing with [28, 39], the major difference for the signer's running time is the use of the inner product argument in [28, 39] and the use of NISA in our scheme. As discussed in the section of NISA, we only use half of the exponentiation used in the inner product argument. Verification time for out scheme is dominated by Line 17 of Algorithm 4, which contains $n + 2\log n + 1$ exponentiations for a ring size of n. [28, 39] used Bulletproof which contains $2n + 2\log n + 1$ exponentiations in verification. To conclude, our DualRing-EC outperforms [11, 23, 28, 39] in terms of signature size and the running time of the signer and the verifier.



(a) Running time of Sign.



(b) Running time of Verify.

Fig. 4: Running times of DualRing-EC

7 DualRing-LB: Our Lattice-Based Ring Signature

In this section, we give a concrete ring signature construction based on standard (module) lattice assumptions using DualRing. As a slight improvement over the conference version of this work, we describe in Appendix D how the construction here can be optimized using the technique from Section 5.4.

Notations and Assumptions. We define q as an odd modulus and R_q as a ring $\mathbb{Z}_q[X]/(X^d+1)$ of dimension d. Define $\boldsymbol{I}_n$ as the identity matrix with size n, $\mathfrak{U}_k$ as a set of polynomials in $\mathbb{Z}[X]/(X^d + 1)$ with infinity norm at most $k \in \mathbb{Z}^+$, and $\mathcal{U}$ as the uniform distribution. The Euclidean $\|\cdot\|$ and infinity $\|\cdot\|_\infty$ norms of a polynomial (or a vector of polynomials) are defined in the standard fashion w.r.t. the coefficient vector of the polynomial. Define the following challenge space:

$$\mathcal{C} = \{\, c \in \mathbb{Z}[X]/(X^d + 1) \,:\, \|c\|_\infty = 1 \,\}. \tag{3}$$

Observe that $|\mathcal{C}| = 3^d$. That is, for $d = 128$, we have $|\mathcal{C}| = 3^{128} > 2^{202}$.

We review the hardness of Module-SIS (M-SIS) (defined in "Hermite normal form" as in [5]) and Module-LWE (M-LWE) problems [19].

Definition 7 (M-SIS$_{n,m,q,\beta_{SIS}}$ Assumption). *For all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}\boldsymbol{A}' \leftarrow_s \mathcal{U}(R_q^{n\times(m-n)}),\\ \boldsymbol{A} = [\boldsymbol{I}_n \| \boldsymbol{A}'], \boldsymbol{z} \leftarrow \mathcal{A}(\boldsymbol{A})\end{array} : \begin{array}{l}\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \in R_q^n,\\ 0 < \|\boldsymbol{z}\| \le \beta_{SIS}\end{array}\right] \le \mathsf{negl}(\lambda).$$

Definition 8 (M-LWE$_{n,m,q,\chi}$ Assumption). *Let χ be a distribution over R_q and $\boldsymbol{s} \leftarrow_s \chi^n$ be a secret key. Define $LWE_{q,\boldsymbol{s}}$ as the distribution obtained by sampling $\boldsymbol{a} \leftarrow_s R_q^n, e \leftarrow_s \chi$ and outputting $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$. For all PPT adversaries $\mathcal{A}$, the probability of distinguishing between m samples from $LWE_{q,\boldsymbol{s}}$ and $\mathcal{U}(R_q^n, R_q)$ is $\mathsf{negl}(\lambda)$.*

Algorithm 6: Lattice-based Type-T* Canonical Identification

```
Procedure Setup($\lambda$):
    set M-LWE parameters $k, m, d, q$;
    define a hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{C}$;
    pick $\boldsymbol{G}' \leftarrow R_q^{k\times(m-k)}$;
    $\boldsymbol{G} = [\boldsymbol{I}_k \,\|\, \boldsymbol{G}']$;
    return $\mathsf{param} = (k, m, d, q, \boldsymbol{G}, \mathcal{H})$;
Procedure KeyGen():
    pick $\boldsymbol{x} \leftarrow \mathfrak{U}_1^m$ ;
    compute $\boldsymbol{c} = \boldsymbol{G}\cdot\boldsymbol{x}$;
    return $(\mathsf{pk}, \mathsf{sk}) = (\boldsymbol{c}, \boldsymbol{x})$;
Procedure Proof1(sk):
    pick $\boldsymbol{r} \leftarrow \mathfrak{U}_{md^2}^m$;
    $\boldsymbol{R} = A(\mathsf{sk}; \boldsymbol{r}) := \boldsymbol{G}\cdot\boldsymbol{r}$;
    return $(\boldsymbol{R}, \boldsymbol{r})$;
Procedure Ch($\boldsymbol{R}$):
    pick $c \leftarrow \mathcal{C}$;
    return $c$;
Procedure Proof2($\mathsf{sk}, \boldsymbol{r}, c$):
    $\boldsymbol{z} = Z(\mathsf{sk}, r, c) := c\cdot\mathsf{sk} - \boldsymbol{r}$;
    if $\|\boldsymbol{z}\|_\infty > md^2 - d$ then
        restart Proof1;
    return $\boldsymbol{z}$;
Procedure Verify($\mathsf{pk}, \boldsymbol{z}, c$):
    $\boldsymbol{R}' = V_1(\boldsymbol{z}) + V_2(\mathsf{pk}, c) := -\boldsymbol{G}\cdot\boldsymbol{z} + c\cdot\mathsf{pk}$;
    if $c \neq$ Ch($\boldsymbol{R}'$) then
        return 0;
    if $\|\boldsymbol{z}\|_\infty > md^2 - d$ then
        return 0;
    return 1;
```

7.1 Lattice-based Canonical Identification

We give a Type-T* canonical identification from M-LWE/SIS in Algorithm 6. We use the rejection sampling technique from [32] to make sure that no information about the signer's secret key is revealed in the response.

We can observe the following

1. The function V_1 is additively homomorphic:
$$V_1(\boldsymbol{z}_1) + V_1(\boldsymbol{z}_2) = -\boldsymbol{G}\cdot\boldsymbol{z}_1 - \boldsymbol{G}\cdot\boldsymbol{z}_2 = -\boldsymbol{G}\cdot(\boldsymbol{z}_1+\boldsymbol{z}_2) = V_1(\boldsymbol{z}_1+\boldsymbol{z}_2).$$
2. Given sk, pk and c, we can compute $\tilde{\boldsymbol{z}} = -c\cdot\mathsf{sk}$ such that $V_1(\tilde{\boldsymbol{z}}) = \boldsymbol{G}\cdot(c\cdot\mathsf{sk}) = V_2(\mathsf{pk}, c)$.
3. The challenge space $\mathcal{C}$ is a group under addition mod 3.

Theorem 8. *Algorithm 6 is secure against special impersonation under key only attack if M-SIS$_{k,m+1,q,\beta_{\mathrm{SIS}}}$ (in HNF) for $\beta_{\mathrm{SIS}} \approx 2d^2\sqrt{m}\cdot\left(1+m\sqrt{d}\right)$ and M-LWE$_{m-k,k,q,\mathfrak{U}_1}$ are hard.*

Proof. Suppose that $\mathcal{A}$ is an adversary breaking the special impersonation under key only attack. Suppose that $\mathcal{B}$ is given $\hat{\boldsymbol{G}} = \left[\boldsymbol{I}_k \,\|\, \boldsymbol{G}' \,\|\, \boldsymbol{g}\right] \in R_q^{k\times(m+1)}$ as the M-SIS matrix where $\boldsymbol{G}'$ and $\boldsymbol{g}$ are sampled uniformly at random. Denote $\boldsymbol{G} = \left[\boldsymbol{I}_k \,\|\, \boldsymbol{G}'\right]$, which is used as the commitment key in the oracle simulations by $\mathcal{B}$. The number of public keys generated by the challenger is q_k. $\mathcal{B}$ sets
$$\mathsf{pk} = \boldsymbol{G}\cdot\boldsymbol{r} + \boldsymbol{g} \tag{4}$$
for $\boldsymbol{r} \leftarrow \mathfrak{U}_1^m$. Observe that $\|\boldsymbol{r}'\| \le \sqrt{md+1}$ for $\boldsymbol{r}' = \begin{pmatrix}\boldsymbol{r}\\1\end{pmatrix}$. Also, note that we can write $\boldsymbol{G}\cdot\boldsymbol{r} = \boldsymbol{r}_0 + \boldsymbol{G}'\cdot\boldsymbol{r}_1$ for $\boldsymbol{r}_0 \in \mathfrak{U}_1^k$ and $\boldsymbol{r}_1 \in \mathfrak{U}_1^{m-k}$. Therefore, by M-LWE$_{m-k,k,q,\mathfrak{U}_1}$ assumption, $\boldsymbol{G}\cdot\boldsymbol{r}$ is computationally indistinguishable from a random element in R_q^k and so is $\mathsf{pk} = \boldsymbol{G}\cdot\boldsymbol{r}+\boldsymbol{g}$. $\mathcal{B}$ gives $\mathsf{param} = (k, m, d, q, \boldsymbol{G}, \mathcal{H})$ and pk to $\mathcal{A}$.

$\mathcal{A}$ returns $(c, \boldsymbol{z}, c', \boldsymbol{z}')$, where $c \neq c'$, we have:
$$-\boldsymbol{G}\cdot\boldsymbol{z} + c\cdot\mathsf{pk} = -\boldsymbol{G}\cdot\boldsymbol{z}' + c'\cdot\mathsf{pk}$$
$$(c-c')\cdot\mathsf{pk} = \boldsymbol{G}\cdot(\boldsymbol{z}-\boldsymbol{z}') = \hat{\boldsymbol{G}}\cdot\begin{pmatrix}\boldsymbol{z}-\boldsymbol{z}'\\0\end{pmatrix}$$

Further, multiplying Eq. (4) by $(c - c')$, we have

$$(c - c') \cdot \mathsf{pk} = \boldsymbol{G} \cdot (c - c') \cdot \boldsymbol{r} + (c - c') \cdot \boldsymbol{g} = \hat{\boldsymbol{G}} \cdot (c - c') \cdot \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix}.$$

Therefore, we get:

$$\hat{\boldsymbol{G}} \cdot (c - c') \cdot \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix} = \hat{\boldsymbol{G}} \cdot \begin{pmatrix} \boldsymbol{z} - \boldsymbol{z}' \\ 0 \end{pmatrix}.$$

That is, $\hat{\boldsymbol{G}} \cdot \boldsymbol{s} = 0$ over R_q for $\boldsymbol{s} = (c - c') \cdot \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix} - \begin{pmatrix} \boldsymbol{z} - \boldsymbol{z}' \\ 0 \end{pmatrix}$. Observe that $\boldsymbol{s}$ cannot be the zero vector as $c \neq c'$ and the last coordinate of $\boldsymbol{s}$ is $(c - c')$. Since $\|\boldsymbol{z}\|_\infty, \|\boldsymbol{z}'\|_\infty \leq md^2 - d$, we also have

$$\|\boldsymbol{s}\| \leq 2d\sqrt{d}\sqrt{md+1} + 2 \cdot (md^2\sqrt{md}) \approx 2d^2\sqrt{m} \cdot \left(1 + m\sqrt{d}\right).$$

Hence, $\boldsymbol{s}$ is a solution to M-SIS$_{k,m+1,q,\beta_{\text{SIS}}}$ for $\beta_{\text{SIS}} \approx 2d^2\sqrt{m}\left(1 + m\sqrt{d}\right)$. □

Remark. It is not known how to build an efficient lattice-based ZK proof for sum argument. There is a theoretical work on constructing a lattice analog of Bulletproofs in [13]. However, in practice, the construction is inefficient. As the lattice analog of the Sum Argument cannot be constructed efficiently, the signature size of our lattice-based construction remains at $O(n)$, while [10,19] achieve $O(\log n)$ signature size. Hence, after some point, our construction eventually produces longer signatures.

7.2 Efficiency Analysis of DualRing-LB

Signature Size. The practical security estimations of M-SIS and M-LWE against known attacks are done by following the methodology detailed in [18, Section 3.2.4]. In particular, we aim for a "root Hermite factor" of around 1.0045. The root Hermite factor is a common metric used in lattice-based cryptography to measure practical hardness. We refer to [18] for further discussion. We refer to Table 3 for the concrete parameter setting. In general, for $d = 128$, the signature length can be approximated by the following formula:

$$|\sigma| = |\boldsymbol{z}| + n \cdot |c_i| \approx 4536 + 26n \text{ bytes.} \tag{5}$$

The above formula stems from the fact that $|c_i| = d\log 3/8$ bytes and $|\boldsymbol{z}| = md\log(2md^2)/8$ bytes since $\boldsymbol{z} \in R^m$ with $\|\boldsymbol{z}\|_\infty \leq md^2$. Plugging in $(d, m) = (128, 15)$ yields (5).

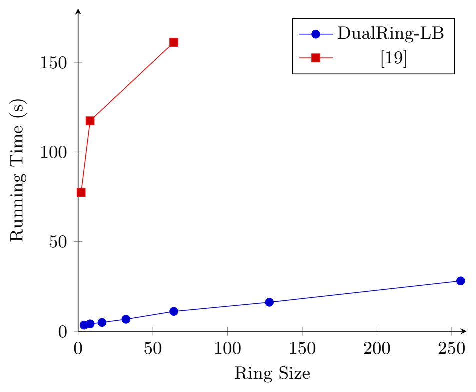
Although Theorems 3 and 8 imply that DualRing-LB is secure, they do not provide all the information required in the concrete parameter setting. Unlike the classical DL- or factoring-based constructions, in the lattice setting, it is important for the concrete parameter setting to know the precise (Euclidean) norm bound β_{SIS} of M-SIS solution that arises in the security reduction. This is because the practical security estimations depend on the β_{SIS} parameter of the M-SIS problem. Therefore, we also need to investigate in more detail the M-SIS solution length β_{SIS} for the *ring signature* (not the underlying Type-T* canonical identification as in Theorem 8) and see how it depends on the parameters. We do this in the full version of the paper and show concretely what the length of the M-SIS solution is for the ring signature, which gives $\beta_{\text{SIS}} \approx 2d\sqrt{md} \cdot (md + n)$. The proof follows the same blueprint in the generic unforgeability proof of DualRing (Theorem 3), but we keep track of the norms as the proof proceeds.

Computational Efficiency. First, the modulus q is always less than 32 bits in length for the parameters in Table 3. Therefore the values in our construction fit into 32-bit registers, boosting the computational efficiency. Another advantage of our construction is that no (discrete) Gaussian sampling is required, making the implementation easier to protect against side-channel attacks.

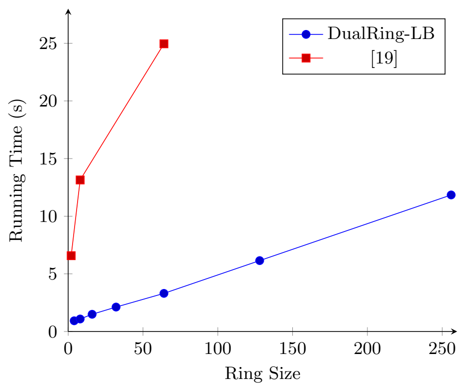
We show the running times of DualRing-LB in Fig. 5. The code is written in Python, using the polynomial arithmetic and NTT transform in the sympy library. It is tested on a computer

n	k	m	$\log q$	Signature Size	PK Size	SK Size	Size of $(c_1, \ldots, c_n)$	Size of $\boldsymbol{z}$
2	7	15	26	4.48	2.84	0.23	0.05	4.43
4	7	15	26	4.53	2.84	0.23	0.10	4.43
8	7	15	26	4.63	2.84	0.23	0.20	4.43
16	7	15	26	4.83	2.84	0.23	0.40	4.43
32	7	15	26	5.22	2.84	0.23	0.79	4.43
64	7	15	26	6.02	2.84	0.23	1.59	4.43
128	7	15	26	7.60	2.84	0.23	3.17	4.43
256	7	15	26	10.78	2.84	0.23	6.34	4.43
512	8	16	26	17.44	3.25	0.25	12.69	4.75
1024	8	16	26	30.13	3.25	0.25	25.38	4.75
2048	8	16	26	55.50	3.25	0.25	50.75	4.75
4096	8	17	27	106.57	3.38	0.27	101.50	5.07

Table 3: The parameter setting of DualRing-LB. The root Hermite factor for both M-SIS and M-LWE are ≤ 1.0045. $d = 128$ always. The sizes are in KB.



(a) Running Time of Sign.



(b) Running Time of Verify.

Fig. 5: Lattice-based ring signatures

with Intel Core i5 2.3GHz, 8GB RAM with MacOS 10. For our scheme, the expected number of iterations due to rejection sampling in Sign is about 2.72 and our experiment matches this prediction. The running time for a single run of sign and verify algorithms are about the same. However, the expected number of iterations for sign is 2.72. Therefore, we have the running time for sign as in Fig. 5.

The construction in [19] is at least 5 times slower than DualRing-LB for both sign and verify. Some of the possible reasons include: (1) their expected number of iterations due to rejection sampling in Sign is about 4.757, (2) they use a polynomial of degree $d = 256$. Their scheme does not exhibit a linear increase in running time since [19] changes the system parameters (e.g., matrix dimension, degree of polynomial) for different ring size to optimize their signature size.

8 Conclusion

In this paper, we propose a generic construction of ring signature scheme using a dual ring structure. When we instantiate in the DL-setting, it is the shortest ring signature scheme without using trusted setup. When instantiated in M-LWE/SIS, we have the shortest ring signature for ring size between 4 and 2000.

Acknowledgment

We thank Vadim Lyubashevsky and Ngoc Khanh Nguyen for pointing out the “trick” in Section 5.4.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Necessary and sufficient conditions for security and forward-security. IEEE Trans. Information Theory **54**(8), 3631–3646 (2008)
2. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer (2002)
3. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setup - from standard assumptions. In: EUROCRYPT 2019. LNCS, vol. 11478, pp. 281–311. Springer (2019)
4. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: CCS 2008. pp. 449–458. ACM (2008)
5. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: SCN 2018. LNCS, vol. 11035, pp. 368–385. Springer (2018)
6. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer (2004)
7. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS 2006. pp. 390–399. ACM (2006)
8. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. J. Cryptology **22**(1), 114–138 (2009)
9. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.: High-speed high-security signatures. In: CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer (2011)
10. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: ASIACRYPT (2). LNCS, vol. 12492, pp. 464–492. Springer (2020)
11. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: ESORICS 2015. LNCS, vol. 9326, pp. 243–265. Springer (2015)
12. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer (2016)
13. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: A non-pcp approach to succinct quantum-safe zero-knowledge. In: CRYPTO 2020. LNCS, vol. 12171, pp. 441–469. Springer (2020)
14. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334 (2018)
15. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO ’92. LNCS, vol. 740, pp. 89–105. Springer (1992)
16. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO ’94. Lecture Notes in Computer Science, vol. 839, pp. 174–187. Springer (1994)
17. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer (2004)
18. Esgin, M.F.: Practice-Oriented Techniques in Lattice-Based Cryptography. Ph.D. thesis, Monash University (5 2020). https://doi.org/10.26180/5eb8f525b3562
19. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: CRYPTO 2019. LNCS, vol. 11692, pp. 115–146. Springer (2019)
20. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: ACNS 2019. LNCS, vol. 11464, pp. 67–88. Springer (2019)
21. Esgin, M.F., Steinfeld, R., Zhao, R.K.: MatRiCT^{+}: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545 (2021), `ia.cr/2021/545` (to appear at IEEE S&P 2022)
22. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In: ACM CCS. pp. 567–584. ACM (2019), (Full version at `ia.cr/2019/1287`)

23. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT 2015. pp. 253–280. LNCS, Springer (2015)
24. Guillou, L.C., Quisquater, J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: CRYPTO '88. LNCS, vol. 403, pp. 216–231. Springer (1988)
25. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: CCS 2003. pp. 155–164. ACM (2003)
26. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: CRYPTO 2016. LNCS, vol. 9815, pp. 33–61. Springer (2016)
27. Kumar, A., Fischer, C., Tople, S., Saxena, P.: A traceability analysis of monero's blockchain. In: ESORICS (2). LNCS, vol. 10493, pp. 153–173. Springer (2017)
28. Lai, R.W.F., Ronge, V., Ruffing, T., Schröder, D., Thyagarajan, S.A.K., Wang, J.: Omniring: Scaling private payments without trusted setup. In: CCS 2019. pp. 31–48. ACM (2019)
29. 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: EUROCRYPT 2016. LNCS, vol. 9666, pp. 1–31. Springer (2016)
30. Libert, B., Peters, T., Qian, C.: Logarithmic-size ring signatures with tight security from the DDH assumption. In: ESORICS 2018. LNCS, vol. 11099, pp. 288–308. Springer (2018)
31. Lu, X., Au, M.H., Zhang, Z.: Raptor: A practical lattice-based (linkable) ring signature. In: ACNS 2019. LNCS, vol. 11464, pp. 110–130. Springer (2019)
32. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer (2009)
33. Lyubashevsky, V.: Lattice signatures without trapdoors. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer (2012)
34. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. In: CRYPTO 2021. LNCS, vol. 12826, pp. 611–640. Springer (2021)
35. Maxwell, G., Poelstra, A.: Borromean ring signatures (2015), https://pdfs.semanticscholar.org/4160/470c7f6cf05ffc81a98e8fd67fb0c84836ea.pdf.
36. Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., Christin, N.: An empirical analysis of traceability in the monero blockchain. PoPETs **2018**(3), 143–163 (2018)
37. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer (2001)
38. Yuen, T.H., Esgin, M.F., Liu, J.K., Au, M.H., Ding, Z.: Dualring: Generic construction of ring signatures with efficient instantiations. In: CRYPTO 2021. LNCS, vol. 12825, pp. 251–281. Springer (2021)
39. Yuen, T.H., Sun, S., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In: FC 2020. LNCS, vol. 12059, pp. 464–483. Springer (2020)

A Type-T AOS Ring Signature and its security analysis

We denote T as a Type-T signature. We also denote Δ_z as the domain of z. The AOS ring signature [2] is summarized in Algorithm 7. There is no security proof for the generic construction of AOS ring signature in [2]. The instantiation of AOS ring signature with Schnorr signature is proven secure in [2].

A.1 Formal Security Proof for Type-T AOS Ring Signature

We changed the definition of AOS Ring Signature from Type-T Signature to Canonical Identification as shown in Algorithm 8. We can prove the security of the (modified) AOS ring signature if the Canonical Identification has special soundness.

Theorem 9. *The AOS ring signature is unforgeable if T is secure against key recovery under key only attack and T has special soundness.*

Algorithm 7: AOS Ring Signature from Type-T Signature T

```
Procedure Setup($\lambda$):
    return param $\leftarrow$ T.Setup($\lambda$);
Procedure KeyGen():
    return (pk, sk) $\leftarrow$ T.KeyGen();
Procedure Sign($m, \mathbf{pk} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_n\}, \mathsf{sk}_j$):
    $r_j \leftarrow_s \Delta_r$;
    $R_j = A(\mathsf{sk}_j; r_j)$;
    for $i \leftarrow j+1, \dots, n, 1, \dots j-1$ do
        $c_i = H(m, \mathbf{pk}, R_{i-1})$;
        $z_i \leftarrow_s \Delta_z$;
        $R_i = V(\mathsf{pk}_i, z_i, c_i)$ ;
        /* Define $R_0 = R_n$ */
    $c_j = H(m, \mathbf{pk}, R_{j-1})$;
    $z_j = Z(\mathsf{sk}_j, r_j, c_j)$;
    return $\sigma = (c_1, z_1, \dots, z_n)$;
Procedure Verify($m, \mathbf{pk} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_n\}, \sigma$):
    parse $\sigma = (c_1, z_1, \dots, z_n)$;
    $R_1 = V(\mathsf{pk}_1, z_1, c_1)$;
    for $i \leftarrow 2$ to $n$ do
        $c_i = H(m, \mathbf{pk}, R_{i-1})$;
        $R_i = V(\mathsf{pk}_i, z_i, c_i)$;
    if $c_1 \neq H(m, \mathbf{pk}, R_n)$ then
        return 0;
    return 1;
```

Algorithm 8: AOS Ring Signature from Canonical Identification T

```
Procedure Setup($\lambda$):
    define $H : \{0,1\}^* \rightarrow \Delta_c$;
    return param $\leftarrow$ T.Setup($\lambda$);
Procedure KeyGen():
    return (pk, sk) $\leftarrow$ T.KeyGen();
Procedure Sign($m, \mathbf{pk} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_n\}, \mathsf{sk}_j$):
    $(R_j, r_j) \leftarrow$ T.Proof1($\mathsf{sk}_j$);
    for $i \leftarrow j+1, \dots, n, 1, \dots j-1$ do
        $c_i = H(m, \mathbf{pk}, R_{i-1})$;
        $z_i \leftarrow_s \Delta_z$;
        $R_i = V(\mathsf{pk}_i, z_i, c_i)$ // Define $R_0 = R_n$
    $c_j = H(m, \mathbf{pk}, R_{j-1})$;
    $z_j =$ T.Proof2($\mathsf{sk}_j, r_j, c_j$);
    return $\sigma = (c_1, z_1, \dots, z_n)$;
Procedure Verify($m, \mathbf{pk} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_n\}, \sigma$):
    parse $\sigma = (c_1, z_1, \dots, z_n)$;
    $R_1 = V(\mathsf{pk}_1, z_1, c_1)$;
    for $i \leftarrow 2$ to $n$ do
        $c_i = H(m, \mathbf{pk}, R_{i-1})$;
        $R_i = V(\mathsf{pk}_i, z_i, c_i)$;
    if $c_1 \neq H(m, \mathbf{pk}, R_n)$ then
        return 0;
    return 1;
```

Proof. Denote $\mathcal{A}$ as a PPT adversary breaking the unforgeability of the AOS ring signature. We build an algorithm $\mathcal{B}$ to break the key recovery under active attack of T. Suppose the algorithm $\mathcal{B}$ is given the system parameters param and the public key pk^* from its challenger $\mathcal{C}_T$.

Setup. $\mathcal{B}$ picks a random index $i^* \in [1, q_k]$. $\mathcal{B}$ runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}()$ for $i \in [1, q_k], i \neq i^*$. $\mathcal{B}$ sets $\mathsf{pk}_{i^*} = \mathsf{pk}^*$. $\mathcal{C}$ gives param and $S := \{\mathsf{pk}_i\}_{i=1}^{q_k}$ to $\mathcal{A}$. $\mathcal{B}$ maintains an initially empty list $\mathcal{H}$.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

– H: $\mathcal{B}$ simulates H as a random oracle. On the k-th distinct query with input $(m, \mathbf{pk}, R)$, $\mathcal{B}$ picks a random $c \in \Delta_c$, puts $(k, m, \mathbf{pk}, R, c)$ in $\mathcal{H}$ and returns c.
– $\mathcal{CO}$: On input i, $\mathcal{B}$ returns sk_i (if $i = i^*$, $\mathcal{B}$ declares failure and exits).
– $\mathcal{SO}$: On input a message m, a set of public key $\mathbf{pk}$ with the signer index j, if $j \neq i^*$, then $\mathcal{B}$ returns $\sigma \leftarrow$ Sign(param, $m, \mathbf{pk}, \mathsf{sk}_j$).
  Otherwise, $\mathcal{B}$ picks random $c_1, z_1, \dots, z_n \in \mathbb{Z}_p$ (such that $(\cdot,\cdot,\cdot,\cdot,c_1) \notin \mathcal{H}$) and computes $R_i = V(\mathsf{pk}_i, z_i, c_i)$ and $c_{i+1} = H(m, \mathbf{pk}, R_i)$, for $i = 1, \dots, n$. $\mathcal{B}$ sets $H(m, \mathbf{pk}, R_n) = c_1$ by the random oracle H. (If there exists $(\cdot, m, \mathbf{pk}, R_n, \cdot) \in \mathcal{H}$, then $\mathcal{B}$ picks random $c_1, z_1, \dots, z_n \in \mathbb{Z}_p$ again.) $\mathcal{B}$ returns $\sigma = (c_1, z_1, \dots, z_n)$.

Challenge. $\mathcal{A}$ returns a forgery $(m^*, \mathbf{pk}^*, \sigma^* = (c_1^*, z_1^*, \ldots, z_n^*))$. Denote $\mathbf{pk}^* = \{\mathsf{pk}_1, \ldots \mathsf{pk}_n\}$. If $\mathsf{pk}^* \neq \mathsf{pk}_{j^*}$ for any $j^* \in [1, n]$, $\mathcal{B}$ declares failure and exits. Otherwise, denote $\mathsf{pk}^* = \mathsf{pk}_{j^*}$.

$\mathcal{B}$ computes $R_i^* = V(\mathsf{pk}_i, z_i^*, c_i^*)$ and $c_{i+1}^* = H(m^*, \mathbf{pk}^*, R_i^*)$, for $i = 1, \ldots, n$. Note that for any index $i \in [1, n]$, $(\cdot, m^*, \mathbf{pk}^*, R_i^*, c_{i+1}^*) \notin \mathcal{H}$ has probability $\frac{1}{|\Delta_c|}$. With probability at least $(1 - \frac{1}{|\Delta_c|})^n \geq 1 - \frac{n}{|\Delta_c|}$, $(\cdot, m^*, \mathbf{pk}^*, R_i^*, c_{i+1}^*) \in \mathcal{H}$ for all i. In this case, there exists at least one index i such that $(k_i, m^*, \mathbf{pk}^*, R_{i-1}^*, c_i^*) \in \mathcal{H}$ and $(k_{i+1}, m^*, \mathbf{pk}^*, R_i^*, c_{i+1}^*) \in \mathcal{H}$ such that $k_i > k_{i+1}$. We call this type of index a *reverse index*. Reverse index exists because these H queries form a ring. With probability at least $1/n$, the index j^* is a reverse index. $\mathcal{B}$ rewinds to the point that $(m^*, \mathbf{pk}^*, R_{j^*-1}^*)$ is queried to H and returns a different c'_{j^*} instead. $\mathcal{A}$ returns another signature $\sigma' = (c'_1, z'_1, \ldots, z'_n)$. Since j^* is a reverse index, $R_{j^*}^*$ remains the same for σ^* and σ'. Since both σ^* and σ' are valid signatures, We have:

$$R_{j^*}^* = V(\mathsf{pk}_{j^*}, z_{j^*}^*, c_{j^*}^*) = V(\mathsf{pk}_{j^*}, z'_{j^*}, c'_{j^*}).$$

Therefore, we have two accepting transcripts $(R_{j^*}^*, c_{j^*}^*, z_{j^*}^*)$ and $(R_{j^*}^*, c'_{j^*}, z'_{j^*})$ with $c_{j^*}^* \neq c'_{j^*}$. By the special soundness property of T, there exists an extractor Ext which can output sk^* with respect to pk^*. $\mathcal{B}$ uses sk^* to break the key recovery under key only attack.

Theorem 10. *The AOS ring signature is anonymous in the random oracle model.*

Proof. We show how to build an algorithm $\mathcal{B}$ providing perfect anonymity in the random oracle model.

Setup. $\mathcal{B}$ runs $\mathsf{param} \leftarrow \textsc{Setup}(\lambda)$. $\mathcal{B}$ runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param}; \omega_i)$ for $i \in [1, q_k]$ with randomness ω_i. $\mathcal{B}$ gives param and $S := \{\mathsf{pk}_i\}_{i=1}^{q_k}$ to $\mathcal{A}_1$.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

- $\mathcal{SO}$: On input a message m, a set of public key $\mathbf{pk}$ with the signer index j, $\mathcal{B}$ returns $\sigma \leftarrow \textsc{Sign}(\mathsf{param}, m, \mathbf{pk}, \mathsf{sk}_j)$.
- H: $\mathcal{B}$ simulates H as a random oracle.

Challenge. $\mathcal{A}_1$ gives $\mathcal{B}$ a message m and a vector of public keys $\mathbf{pk}$ and two indices i_0, i_1. $\mathcal{B}$ picks random $c_1, z_1, \ldots, z_n \in \mathbb{Z}_p$ and computes $R_i = V(\mathsf{pk}_i, z_i, c_i)$ and $c_{i+1} = H(m, \mathbf{pk}, R_i)$, for $i = 1, \ldots, n$. $\mathcal{B}$ sets $H(m, \mathbf{pk}, R_n) = c_1$ in the random oracle. (If the hash value is already set by the H oracle, $\mathcal{B}$ picks another random $c_1, z_1, \ldots, z_n \in \mathbb{Z}_p$ and tries again.) $\mathcal{B}$ returns $\sigma = (c_1, z_1, \ldots, z_n)$ and $\{\omega_i\}_{i=1}^{q_k}$ to $\mathcal{A}_2$. $\mathcal{B}$ picks a random bit b.

Output. Finally, $\mathcal{A}_2$ outputs a bit b'. Observe that b is not used in the generation of σ. Therefore, $\mathcal{A}_2$ can only win with probability 1/2.

B Security Proof for Sum Argument

To show the security of our NISA protocol, we define a new relation:

$$\left\{(\mathbf{g} \in \mathbb{G}^n, u \in \mathbb{G}, P \in \mathbb{G}, \mathbf{b} = \mathbf{1}^n; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} u^{\langle \mathbf{a}, \mathbf{b} \rangle}\right\} \tag{6}$$

Not only we set the vector $\mathbf{b} = \mathbf{1}^n$, we also remove the vector of generators $\mathbf{h}$ corresponding to $\mathbf{b}$ to further improve the efficiency. It is because we do not need to prove the public vector of scalars $\mathbf{b}$ in our sum argument.

First, we give Algorithm 9 for Relation 2. Here, we note that the DL relation between generators $\mathbf{g}$ and u is unknown by $\mathcal{P}$. In Algorithm 9, $\mathcal{V}$ chooses a x to $\mathcal{P}$, ensuring that $\sum \mathbf{a} = c$.

To complete Algorithm 9, we propose Algorithm 10. Algorithm 10 shows how to construct a sum argument based on an inner product argument from [14]. The main idea of our protocol is to set one of the two vectors of inner product argument $\mathbf{b} = \mathbf{1}^n$. And we omit the multi-exponentiations corresponding to $\mathbf{b}$ since both $\mathcal{P}$ and $\mathcal{V}$ know its value. As we can see, length of

Algorithm 9: Interactive proof system $(\mathcal{P}, \mathcal{V})$ for Relation 2

```
input: $(\mathbf{g} \in \mathbb{G}^n, u, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a} \in \mathbb{Z}_p^n)$;
input of $\mathcal{P} : (\mathbf{g}, u, P, c, \mathbf{a})$,     input of $\mathcal{V} : (\mathbf{g}, u, P, c)$;
1 $\mathcal{V} : x \leftarrow \mathbb{Z}_p^\star$;
2 $\mathcal{V} \rightarrow \mathcal{P} : x$;
3 $\mathcal{P}$ and $\mathcal{V}$ computes:$P' = P \cdot u^{x \cdot c}$;
4 run Algorithm 10 on input $(\mathbf{g}, u^x, P', \mathbf{1}^n; \mathbf{a})$;
```

Algorithm 10: Interactive proof system $(\mathcal{P}, \mathcal{V})$ for Relation 6

```
input: $(\mathbf{g} \in \mathbb{G}^n, u, P \in \mathbb{G}, \mathbf{b} \in \mathbb{Z}_p^n; \mathbf{a} \in \mathbb{Z}_p^n)$;
input of $\mathcal{P} : (\mathbf{g}, u, P, \mathbf{b}, \mathbf{a})$;
input of $\mathcal{V} : (\mathbf{g}, u, P, \mathbf{b})$;
output: {accept or reject};
1  if $n = 1$ then
2      $\mathcal{P} \rightarrow \mathcal{V} : a \in \mathbb{Z}_p$;
3      $\mathcal{V}$ computes $c = b \cdot a$;
4      if $P = g^a u^c$ then
5          $\mathcal{V}$ outputs accept;
6      else
7          $\mathcal{V}$ outputs reject;
8  else
9      $\mathcal{P}$ computes $n' = \frac{n}{2}$, $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$, $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$;
10     $\mathcal{P}$ computes $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} u^{c_L} \in \mathbb{G}$ and $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} u^{c_R} \in \mathbb{G}$;
11     $\mathcal{P} \rightarrow \mathcal{V} : L, R$;
12     $\mathcal{V} \rightarrow \mathcal{P} : x \leftarrow \mathbb{Z}_p$;
13     $\mathcal{P}$ and $\mathcal{V}$ compute:  $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$, $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$, and
       $\mathbf{b}' = x^{-1} \cdot \mathbf{b}_{[:n']} + x \cdot \mathbf{b}_{[n':]} \in \mathbb{Z}_p^{n'}$;
14     $\mathcal{P}$ computes  $\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$;
15     run this protocol on input $(\mathbf{g}', u, P', \mathbf{b}'; \mathbf{a}')$;
```

vector $\mathbf{a}$ is reduced by half in each iteration, thus we get a value a instead of a vector $\mathbf{a}$ in the last iteration. The outputs of each iteration are two scalars (L, R), therefore the communication complexity of our protocol is $O(\log_2(n))$. Moreover, our protocol can become non-interactive by applying the Fiat-Shamir transformation.

Security of Sum Argument. Perfect completeness of the sum argument is straightforward. The proof of Theorem 5 is as follows.

Proof. We prove that there is an efficient extractor $\mathcal{X}$ which uses n^2 transcripts. The argument is trivially sound if $n = 1$ since the witness a is given to the verifier $\mathcal{V}$. Considering each iteration with input $(\mathbf{g}, u, P, \mathbf{b})$ there is an extractor that can easily extract a witness $\mathbf{a}$ or a non-trivial discrete logarithm relation between $\mathbf{g}, u$ from the prover. It has access to the prover to get L and R and can obtain four vectors $\mathbf{a}_i' \in \mathbb{Z}_p^{n'}$ after rewinding the prover three times with three challenges $\{x_i\}_{i \in [3]}$ such that $\{|x_i| \neq |x_j|\}_{1 \leq i < j \leq 3}$. Recall that

$$L^{x_i^2} P R^{x_i^{-2}} = \left(\mathbf{g}_{[:n']}^{x_i^{-1}} \circ \mathbf{g}_{[n':]}^{x_i} \right)^{\mathbf{a}_i'} u^{\langle \mathbf{a}_i', \mathbf{b}_i' \rangle}, i \in [3]. \quad (7)$$

Then we set $\{v_{j,i}\}_{j\in[3],i\in[3]}$ such that

$$\begin{cases}\sum_i v_{1,i}x_i^2 = 1\\ \sum_i v_{1,i} = 0\\ \sum_i v_{1,i}x_i^{-2} = 0\end{cases} \begin{cases}\sum_i v_{2,i}x_i^2 = 0\\ \sum_i v_{2,i} = 1\\ \sum_i v_{2,i}x_i^{-2} = 0\end{cases} \begin{cases}\sum_i v_{3,i}x_i^2 = 0\\ \sum_i v_{3,i} = 0\\ \sum_i v_{3,i}x_i^{-2} = 1\end{cases}$$

We raise the three equalities of (7) to powers $\{v_{1,i}\}_{i\in[3]}$ such that

$$\begin{aligned}\prod_i \left(L^{x_i^2}PR^{x_i^{-2}}\right)^{v_{1,i}} &= \prod_i \left(\left(\mathbf{g}_{[:n']}^{x_i^{-1}} \circ \mathbf{g}_{[n':]}^{x_i}\right)^{\mathbf{a}'_i} u^{\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle}\right)^{v_{1,i}}\\ L^{\sum_i v_{1,i}x_i^2}P^{\sum_i v_{1,i}}R^{\sum_i v_{1,i}x_i^{-2}} &= \prod_i \left(\mathbf{g}_{[:n']}^{x_i^{-1}\mathbf{a}'_i}\mathbf{g}_{[n':]}^{x_i\mathbf{a}'_i}u^{\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle}\right)^{v_{1,i}}\\ L &= \mathbf{g}_{[:n']}^{\sum_i v_{1,i}x_i^{-1}\mathbf{a}'_i}\mathbf{g}_{[n':]}^{\sum_i v_{1,i}x_i\mathbf{a}'_i}u^{\sum_i v_{1,i}\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle}.\end{aligned}$$

We denote $L = \mathbf{g}^{\mathbf{a}_L}u^{c_L}$ for some $\mathbf{a}_L \in \mathbb{Z}_p^n, c_L \in \mathbb{Z}_p$, such that

$$\mathbf{a}_L = (\sum_i v_{1,i}x_i^{-1}\mathbf{a}'_i, \sum_i v_{1,i}x_i\mathbf{a}'_i), \quad c_L = \sum_i v_{1,i}\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle.$$

Similarly, we can denote $P = \mathbf{g}^{\mathbf{a}_P}u^{c_P}$ and $R = \mathbf{g}^{\mathbf{a}_R}u^{c_R}$ for some $\mathbf{a}_P \in \mathbb{Z}_p^n, c_P \in \mathbb{Z}_p, \mathbf{a}_R \in \mathbb{Z}_p^n, c_R \in \mathbb{Z}_p$, such that

$$\begin{aligned}\mathbf{a}_P &= (\sum_i v_{2,i}x_i^{-1}\mathbf{a}'_i, \sum_i v_{2,i}x_i\mathbf{a}'_i), \quad c_P = \sum_i v_{2,i}\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle,\\ \mathbf{a}_R &= (\sum_i v_{3,i}x_i^{-1}\mathbf{a}'_i, \sum_i v_{3,i}x_i\mathbf{a}'_i), \quad c_R = \sum_i v_{3,i}\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle.\end{aligned}$$

We note that (7) can be described as follows:

$$L^{x_i^2}PR^{x_i^{-2}} = \mathbf{g}_{[:n']}^{x_i^{-1}\mathbf{a}'_i}\mathbf{g}_{[n':]}^{x_i\mathbf{a}'_i}u^{\langle \mathbf{a}'_i,\mathbf{b}'_i\rangle}, i \in [3].$$

By putting back the definition of $L = \mathbf{g}^{\mathbf{a}_L}u^{c_L}$, $R = \mathbf{g}^{\mathbf{a}_R}u^{c_R}$ and $P = \mathbf{g}^{\mathbf{a}_P}u^{c_P}$, we also have:

$$L^{x_i^2}PR^{x_i^{-2}} = \mathbf{g}^{x_i^2\mathbf{a}_L+\mathbf{a}_P+x_i^{-2}\mathbf{a}_R}u^{x_i^2c_L+c_P+x_i^{-2}c_R}, i \in [3].$$

It implies for $i \in [3]$,

$$\begin{aligned}x_i^{-1}\mathbf{a}'_i &= x_i^2\mathbf{a}_{L,[:n']} + \mathbf{a}_{P,[:n']} + x_i^{-2}\mathbf{a}_{R,[:n']},\\ x_i\mathbf{a}'_i &= x_i^2\mathbf{a}_{L,[n':]} + \mathbf{a}_{P,[n':]} + x_i^{-2}\mathbf{a}_{R,[n':]},\\ \langle \mathbf{a}'_i,\mathbf{b}'_i\rangle &= x_i^2c_L + c_P + x_i^{-2}c_R.\end{aligned} \tag{8}$$

Here we can get a non-trivial DL relation among $\mathbf{g}, u$ if any of these equations are not equal. Otherwise we have

$$\begin{cases}\mathbf{a}'_i = x_i^3\mathbf{a}_{L,[:n']} + x_i\mathbf{a}_{P,[:n']} + x_i^{-1}\mathbf{a}_{R,[:n']}\\ \mathbf{a}'_i = x_i\mathbf{a}_{L,[n':]} + x_i^{-1}\mathbf{a}_{P,[n':]} + x_i^{-3}\mathbf{a}_{R,[n':]}\end{cases}, i \in [3]$$

combining the two equations we have

$$0 = x_i^3\mathbf{a}_{L,[:n']} + x_i(\mathbf{a}_{P,[:n']} - \mathbf{a}_{L,[n':]}) + x_i^{-1}(\mathbf{a}_{R,[:n']} - \mathbf{a}_{P,[n':]}) - x_i^{-3}\mathbf{a}_{R,[n':]}, i \in [3].$$

Since the above holds for all x_i,

$$\mathbf{a}_{L,[:n']} = 0, \qquad \mathbf{a}_{R,[n':]} = 0, \qquad \mathbf{a}_{P,[:n']} = \mathbf{a}_{L,[n':]}, \qquad \mathbf{a}_{R,[:n']} = \mathbf{a}_{P,[n':]}.$$

Using this we have

$$\mathbf{a}'_i = x_i \mathbf{a}_{P,[:n']} + x_i^{-1} \mathbf{a}_{P,[n':]}, i \in [3].$$

Then we compute an inner product

$$\begin{aligned}\langle \mathbf{a}'_i, \mathbf{b}'_i \rangle &= \langle x_i \mathbf{a}_{P,[:n']} + x_i^{-1} \mathbf{a}_{P,[n':]}, x_i^{-1} \mathbf{b}_{[:n']} + x_i \mathbf{b}_{[n':]} \rangle \\ &= \langle \mathbf{a}_P, \mathbf{b} \rangle + x_i^2 \langle \mathbf{a}_{P,[:n']}, \mathbf{b}_{[n':]} \rangle + x_i^{-2} \langle \mathbf{a}_{P,[n':]}, \mathbf{b}_{[:n']} \rangle, \quad i \in [3]\end{aligned}$$

where the vector $\mathbf{b}$ is public as we defined in Relation 6. Recall from equation (8) that

$$\langle \mathbf{a}'_i, \mathbf{b}'_i \rangle = x_i^2 c_L + c_P + x_i^{-2} c_R, \quad i \in [3].$$

The above holds for all x_i. Thus we can conclude that the extractor $\mathcal{X}$ extracts either a DL relation among $\mathbf{g}, u$ or a witness $\mathbf{a}_P$ such that $\langle \mathbf{a}_P, \mathbf{b} \rangle = c_P$.

For Algorithm 9, we construct an extractor $\mathcal{X}_s$ which receives two witness $\mathbf{a}_1, \mathbf{a}_2$ using the extractor $\mathcal{X}$ twice with different challenges x_1, x_2. Thus we can compute:

$$u^{(x_1 - x_2)c} = \mathbf{g}^{\mathbf{a}_1 - \mathbf{a}_2} u^{x_1 \langle \mathbf{a}_1, \mathbf{b} \rangle - x_2 \langle \mathbf{a}_2, \mathbf{b} \rangle}.$$

Assuming we cannot find non-trivial DL relation among $\mathbf{g}, u$, we get $c = \langle \mathbf{a}_1, \mathbf{b} \rangle$ since $\mathbf{a}_1 = \mathbf{a}_2$. As a result, we conclude that our protocol has statistical witness-extended emulation.

Non-Interactive Sum Argument. In the Random Oracle Model, it is easy to obtain a non-interactive protocol for sum argument using the Fiat-Shamir heuristic. For Algorithm 9, the prover $\mathcal{P}$ computes $H_Z(\mathbf{g}, P, c)$ to replace x. And for Algorithm 10, $x_j = H_Z(L_j, R_j)$ in each iteration. Correspondingly, the verification should be computed as

$$\mathbf{L}^{\mathbf{x}^2} P \mathbf{R}^{\mathbf{x}^{-2}} \overset{?}{=} g^a u^{a \cdot b},$$

where $(\mathbf{L}, \mathbf{x}, \mathbf{R})$ are generated in $\log_2 n$ iterations.

To accelerate the verification, the verifier $\mathcal{V}$ computes $g = \mathbf{g}^{\mathbf{y}}$ where

$$y_i = \prod_{j \in [\log_2 n]} x_j^{f(i,j)}, \qquad f(i,j) = \begin{cases} 1 & \text{if } (i-1)\text{'s } j\text{-th bit is } 1 \\ -1 & \text{otherwise} \end{cases}$$

The same trick is used when computing b.

C Further Analysis for DualRing-LB

For completeness, let us analyze in more detail how the unforgeability of the ring signature works. The proof still follows the same blueprint in the generic proof of Theorem 3, but we keep track of the norms as in the proof of Theorem 8. We show that if there exists an PPT forger $\mathcal{A}$ who has a non-negligible success probability of creating a successful forgery, then there is a PPT M-SIS solver $\mathcal{B}$ that can break M-SIS$_{k,m+1,q,\beta_{\text{SIS}}}$ (in HNF) with non-negligible probability for $\beta_{\text{SIS}} \approx 2d\sqrt{md} \cdot (md + n)$. We use techniques similar to those in [19, 20] to handle the relaxation in the underlying zero-knowledge proof.

Setup. Suppose that $\mathcal{B}$ is given $\hat{\boldsymbol{G}} = \left[\boldsymbol{I}_k \parallel \boldsymbol{G}' \parallel \boldsymbol{g} \right] \in R_q^{k \times (m+1)}$ as the M-SIS matrix where $\boldsymbol{G}'$ and $\boldsymbol{g}$ are sampled uniformly at random. Denote $\boldsymbol{G} = \left[\boldsymbol{I}_k \parallel \boldsymbol{G}' \right]$, which is used as the commitment key in the oracle simulations by $\mathcal{B}$. The number of public keys generated by the challenger is q_k. $\mathcal{B}$ picks $i^* \leftarrow \{1, \ldots, q_k\}$ and generates a set of public keys $S = \{\hat{\mathsf{pk}}_i\}_{i=1}^{q_k}$ by running $(\hat{\mathsf{sk}}_i, \hat{\mathsf{pk}}_i) \leftarrow \textsc{Keygen}()$ except for index i^*. $\mathcal{B}$ sets

$$\hat{\mathsf{pk}}_{i^*} = \boldsymbol{G} \cdot \boldsymbol{r} + \boldsymbol{g} \tag{9}$$

for $\boldsymbol{r} \leftarrow \mathfrak{U}_1^m$. Observe that

$$\|\boldsymbol{r}'\| \leq \sqrt{md + 1} \qquad \text{for } \boldsymbol{r}' = \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix}. \tag{10}$$

Also, note that we can write $\boldsymbol{G} \cdot \boldsymbol{r} = \boldsymbol{r}_0 + \boldsymbol{G} \cdot \boldsymbol{r}_1$ for $\boldsymbol{r}_0 \in \mathfrak{U}_1^k$ and $\boldsymbol{r}_1 \in \mathfrak{U}_1^{m-k}$. Therefore, by M-LWE$_{m-k,k,q,\mathfrak{U}_1}$ assumption, $\boldsymbol{G} \cdot \boldsymbol{r}$ is computationally indistinguishable from a random element in R_q^k and so is $\hat{\mathsf{pk}}_{i^*} = \boldsymbol{G} \cdot \boldsymbol{r} + \boldsymbol{g}$. Thus, since any public key generated by KEYGEN satisfy the same property by M-LWE$_{m-k,k,q,1}$ assumption, $\hat{\mathsf{pk}}_{i^*}$ is computationally indistinguishable from any other public key. $\mathcal{B}$ gives $\mathsf{param} = (k, m, d, q, \boldsymbol{G}, \mathcal{H})$ and S to $\mathcal{A}$,

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

- $\mathcal{H}$: $\mathcal{B}$ simulates $\mathcal{H}$ as a random oracle.
- $\mathcal{CO}$: On input i, $\mathcal{B}$ returns $\hat{\mathsf{sk}}_i$ (If $i = i^*$, $\mathcal{B}$ declares failure and exits.).
- $\mathcal{SO}$: On input a message M, a set of public key $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and a signer index j, it outputs $\perp$ if $\hat{\mathsf{pk}}_j \notin \mathbf{pk}$. If $j \neq i^*$, then $\mathcal{B}$ returns $\sigma \leftarrow$ SIGN$(\mathsf{param}, M, \mathbf{pk}, \hat{\mathsf{sk}}_j)$.
  Otherwise, $\mathcal{B}$ picks random $\boldsymbol{z}$ with $\|\boldsymbol{z}\|_\infty \le md^2 - d$ and random $c_1, \ldots, c_n \in \mathcal{C}$ and computes $\boldsymbol{R} = \sum_{i=1}^{n} c_i \mathsf{pk}_i - \boldsymbol{G} \cdot \boldsymbol{z}$. $\mathcal{B}$ sets $\mathcal{H}(M, \mathbf{pk}, \boldsymbol{t}) = \sum_{i=1}^n c_i \bmod 3$ in the random oracle. If such value has been set in the random oracle, $\mathcal{B}$ declares failure and exits. $\mathcal{B}$ returns $\sigma = (\boldsymbol{z}, c_1, \ldots, c_n)$.

Challenge. In the end, $\mathcal{A}$ outputs a forgery $(M^*, \{\hat{\mathsf{pk}}_{i_j}\}_{j=1}^n, \sigma = (\boldsymbol{z}, c_1, \ldots, c_n))$. Denote $\mathbf{pk}^* := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n) = (\hat{\mathsf{pk}}_{i_0}, \ldots \hat{\mathsf{pk}}_{i_n})$. If $\hat{\mathsf{pk}}_{i^*} \neq \hat{\mathsf{pk}}_{i_j}$ for all j, $\mathcal{B}$ declares failure and exits. Otherwise, denote $\hat{\mathsf{pk}}_{i^*} = \hat{\mathsf{pk}}_{i_{j^*}}$ for some $j^* \in [1, n]$.

$\mathcal{B}$ computes $\boldsymbol{R} = \sum_{i=1}^{n} c_i \mathsf{pk}_i - \boldsymbol{G} \cdot \boldsymbol{z}$. $\mathcal{B}$ rewinds to the point that $(M^*, \mathbf{pk}^*, \boldsymbol{R})$ is queried to the random oracle and returns a different c' instead. $\mathcal{A}$ outputs another signature $\sigma' = (\boldsymbol{z}', c_1', \ldots, c_n')$ such that $\|\boldsymbol{z}\|_\infty, \|\boldsymbol{z}'\|_\infty \le md^2 - d$ and the following holds

$$\sum_{i=0}^{n-1} c_i \mathsf{pk}_i - \boldsymbol{G} \cdot \boldsymbol{z} = \sum_{i=1}^{n} c_i' \mathsf{pk}_i - \boldsymbol{G} \cdot \boldsymbol{z}'. \tag{11}$$

Since $c_1 + \cdots + c_n \equiv c \pmod 3$ and $c_1' + \cdots + c_n' \equiv c' \pmod 3$ for two *distinct* random oracle outputs $c \neq c'$, there must be a $j^* \in [1, n]$ such that $c_{j^*} \neq c_{j^*}'$. With probability $1/n$, $i^* = i_{j^*}$. We rewrite (11) and obtain

$$\begin{aligned}
\left(c_{j^*} - c_{j^*}'\right) \mathsf{pk}_{j^*} &= \boldsymbol{G} \cdot \boldsymbol{z} - \boldsymbol{G} \cdot \boldsymbol{z}' + \sum_{j=1, i \neq j^*}^{n} \left(c_j' - c_j\right) \mathsf{pk}_j \\
&= \boldsymbol{G} \cdot \left[\boldsymbol{z} - \boldsymbol{z}' + \sum_{j=1, j \neq j^*}^{n} \left(c_j' - c_j\right) \cdot \boldsymbol{r}_j \right] \\
&= \hat{\boldsymbol{G}} \cdot \left[\begin{pmatrix} \boldsymbol{z} - \boldsymbol{z}' \\ 0 \end{pmatrix} + \sum_{j=1, j \neq j^*}^{n} \left(c_j' - c_j\right) \cdot \begin{pmatrix} \boldsymbol{r}_j \\ 0 \end{pmatrix} \right].
\end{aligned} \tag{12}$$

Further, multiplying (9) by $(c_{j^*} - c_{j^*}')$, we have

$$\begin{aligned}
\left(c_{j^*} - c_{j^*}'\right) \hat{\mathsf{pk}}_{i^*} &= \boldsymbol{G} \cdot \left(c_{j^*} - c_{j^*}'\right) \boldsymbol{r} + \left(c_{j^*} - c_{j^*}'\right) \boldsymbol{g} \\
&= \hat{\boldsymbol{G}} \cdot \left(c_{j^*} - c_{j^*}'\right) \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix}.
\end{aligned} \tag{13}$$

Note that $\mathsf{pk}_{j^*} = \hat{\mathsf{pk}}_{i_{j^*}} = \hat{\mathsf{pk}}_{i^*}$. Plugging (13) into (12), we get

$$\hat{\boldsymbol{G}} \cdot \left(c_{j^*} - c_{j^*}'\right) \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix} = \hat{\boldsymbol{G}} \cdot \left[\begin{pmatrix} \boldsymbol{z} - \boldsymbol{z}' \\ 0 \end{pmatrix} + \sum_{j=1, j \neq j^*}^{n} \left(c_j' - c_j\right) \cdot \begin{pmatrix} \boldsymbol{r}_j \\ 0 \end{pmatrix} \right].$$

That is, $\hat{\boldsymbol{G}} \cdot \boldsymbol{s} = \boldsymbol{0}$ over R_q for

$$\boldsymbol{s} = \left(c_{j^*} - c'_{j^*}\right) \begin{pmatrix} \boldsymbol{r} \\ 1 \end{pmatrix} - \left[\begin{pmatrix} \boldsymbol{z} - \boldsymbol{z}' \\ 0 \end{pmatrix} + \sum_{j=1, j \neq j^*}^{n} (c'_j - c_j) \cdot \begin{pmatrix} \boldsymbol{r}_j \\ 0 \end{pmatrix} \right].$$

Observe that $\boldsymbol{s}$ cannot be the zero vector as $c_{j^*} \neq c'_{j^*}$ in that case and the last coordinate of $\boldsymbol{s}$ is $c_{j^*} - c'_{j^*}$. Since $\|\boldsymbol{z}\|_\infty, \|\boldsymbol{z}'\|_\infty \leq md^2 - d$, we also have

$$\begin{aligned} \|\boldsymbol{s}\| &\leq 2d\sqrt{md+1} + 2 \cdot (md^2\sqrt{md}) + (n-1) \cdot \left(2d\sqrt{md}\right) \\ &\approx 2d\sqrt{md} \cdot (md + n)\,. \end{aligned}$$

Therefore, $\boldsymbol{s}$ gives a solution to M-SIS$_{k,m+1,q,\beta_{\text{SIS}}}$ for

$$\beta_{\text{SIS}} \approx 2d\sqrt{md} \cdot (md + n)\,. \tag{14}$$

□

D Optimized DualRing-LB

We describe here how to use the "trick" from Section 5.4 to optimize DualRing-LB slightly. We first choose $\kappa = 192$ as a reasonable choice for the entropy of the challenge space. It is straightforward to adjust it as desired. In particular, some works choose a 128-bit challenge space and $\kappa = 128$ could be assumed to compare with such works. Then, we re-define the challenge space as

$$\mathcal{C} = \{\, c \in \mathbb{Z}[X]/(X^d + 1) \,:\, \|c\|_1 = w \,\wedge\, \|c\|_\infty = 1 \,\}. \tag{15}$$

For example, setting $(d, w) = (256, 39)$ leads to $|\mathcal{C}| > 2^{192}$ and we can just discard some elements to match $|\mathcal{C}| = 2^\kappa = 2^{192}$.

With this, we can now bound the β_{SIS} more tightly as $\beta_{\text{SIS}} \approx 2w\sqrt{md} \cdot (md + n)$ since the ℓ_1-norm of a challenge has dropped from d to w. Then, looking at the concrete parameters, we can set $(d, w) = (256, 39)$, $q \approx 2^{26}$, $k = 3$ and $m = 7$ for $n \leq 2048$. Overall, the ring signature length can then be approximated as

$$|\sigma| = |\boldsymbol{z}| + n \cdot |\hat{c}_i| \approx 3829 + \kappa n/8 \text{ bytes} = 3829 + 24n \text{ bytes}, \tag{16}$$

saving about 700 bytes in the "constant" cost.