




Designated-Verifier Ring Signatures: Strong Definitions, Generic Constructions and Efficient Instantiations

Jiaming Wen^{1,2(✉)}, Willy Susilo², Yanhua Zhang^{2,3}, Fuchun Guo²,
and Huanguo Zhang^{1(✉)}

¹ Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, School of Cyber Science and Engineering,
Wuhan University, Wuhan 430072, China
{wenjm,liss}@whu.edu.cn

² Institute of Cybersecurity and Cryptology,
School of Computing and Information Technology,
University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia
{wsusilo,fuchun}@uow.edu.au, yhzhang@email.zzuli.edu.cn

³ School of Computer Science and Technology,
Zhengzhou University of Light Industry, Zhengzhou 450002, China

Abstract. Ring signatures play crucial roles in cryptographic toolkits, providing the signer with both *anonymity* and *unforgeability*. Over the years, numerous advancements in ring signatures have been developed to address a wide range of application scenarios. However, most of them do not consider that disclosed message-signature pairs are *publicly verifiable*, resulting in a loss of privacy. To tackle this dilemma, we focus on the notion of Designated-Verifier Ring Signatures (DVRS), in which only a designated verifier can ascertain the signature's origin and validity, while others cannot. Unlike directly encrypting a signature to prevent verification, DVRS remain effective even if the designated verifier's secret key is leaked or stolen, enabling enhanced privacy.

After clarifying the necessities and strong definitions of DVRS, we provide a generic construction based on the Extended Type-T* Canonical Identification. This newly defined three-move Canonical Identification can be instantiated from different hardness assumptions. Subsequently, we present two settings. The first one is based on the discrete logarithm (DL) and Decisional Diffie-Hellman (DDH) assumptions. A novel zero-knowledge argument system, Minus Argument, is also devised to reduce signature sizes to logarithmic while not requiring any trusted setup. The second one is based on lattice, which is believed to be quantum-resistant.

Keywords: Designated-Verifier Signature · Ring Signature · Generic Construction · Zero-Knowledge Proof · Post-Quantum Cryptography

1 Introduction

Ring Signatures, introduced at the beginning of this century [21], are a type of signature that allows the signer to authenticate messages while concealing

its identity within an ad hoc set of members chosen by itself (called ring). For the signer, the advantages include no group manager or group secret sharing setup, no joining procedure, etc, and basic security properties encompass both *unforgeability* inherited from digital signatures and new added *anonymity*. The *anonymity* renders it infeasible for anyone, including verifiers, to identify the actual signer within a ring. This enhances the privacy of signer’s identity, making it well-suited for anonymous authentication [16,25].

While enhancing *anonymity*, in scenarios where a signer intends to disclose private information, it becomes necessary to restrict entities that are capable of verifying signatures. A related primitive, known as Designated-Verifier Signatures (DVS) [11], allows the designated verifier to generate a simulated signature that appears indistinguishable from one produced by the original signer. This simulation ensures only the designated verifier can be convinced that a signature indeed from the original signer rather than itself, while others lack the ability to make the distinctions and will not trust the signature. Thus, the designated verifier property is achieved while maintaining *public verifiability*. To distinguish it from later concepts, this property is known as weak designated verifier.

In fact, the above weak designated verifier is often insufficient, necessitating the definition of a strong designated verifier [22,24]. The strong concept aligns more closely with the literal understanding of “designated verifier”, meaning that only the designated verifier can verify signatures, while others cannot. Moreover, even if the designated verifier’s secret key is leaked or stolen, others still cannot validate the signature. It is also the reason why the strong designated verifier property cannot be achieved by simply encrypting a conventional signature. In such cases, if the designated verifier’s secret key is compromised, the signature can be decrypted and then verified.

In ICISC 2021, Behrouz et al. introduced a ring signature scheme incorporating a designated verifier [4], which was slightly improved in a subsequent work [3]. To our knowledge, [8,14,15] also attempt this primitive prior to these studies. However, all the above schemes exhibit three primary drawbacks:

1. **Weak Definition.** They only achieve the weak designated verifier property, while [8,14,15] even lacking rigorous security definitions or proofs.
2. **Increased Sizes.** The signature sizes of them scale linearly with ring sizes.
3. **Pre-Quantum.** They based on assumptions such as discrete logarithm and pairings, rendering them susceptible to the approaching quantum computers.

In this work, we investigate Designated Verifier Ring Signatures (DVRS) with strong designated-verifier property. Concretely, we have the below contributions.

1.1 Our Contributions

We commence with the definitions of strong Designated-Verifier Ring Signatures, and provide a model that encompasses all the security properties that we deem necessary: *unforgeability*, *signer anonymity*, and *non-transferability*. This model for strong DVRS is novel, and also has the potential for applications or

as building blocks. It should be noted that, compared with directly encrypting the signature of a conventional ring signature scheme, the advantage lies in that even if the secret key is leaked or stolen, users other than the designated verifier still cannot validate the signature, providing enhanced privacy and security.

As for constructions, we begin by giving some intuition about the designing for Type-T DVRS, which is non-trivial due to the requirement of the simulation algorithm. To solve the challenge, we develop a new three-move canonical identification named Extended Type-T* Canonical Identification. Then, we minimize signature sizes by leveraging commutative group operations, and integrating the hidden/recovery functions compatible with the optimized construction. Finally, we obtain TripleRing, a generic construction for strong DVRS. We also provide the detailed analysis of its correctness and security.

In terms of instantiations, we present two settings from trendy assumptions. The first is based on DL and DDH, with the merit of requiring only logarithmic signature sizes. As a contrast, previous schemes that achieve only provide the weak designated verifier property require linear sizes, representing a significant reduction. Moreover, the newly devised Minus Argument, may be of independent interest for optimizing other cryptographic protocols such as RingCT. The second is based on lattice, marking the first DVRS scheme that, to the best of our knowledge, is conjectured to be quantum-resistant (including schemes that achieved only the weak designated verifier property).

1.2 Roadmap

The remainder of this paper is organized as follows. Section 2 covers algorithm definitions and security definitions for strong DVRS. Section 3 introduces generic construction while Section 4 provides efficient instantiations with discussions. Finally, Section 5 summarizes the whole paper and gives future prospects.

2 Definitions

2.1 Algorithm Definitions

The strong DVRS scheme consists of five probabilistic polynomial time (PPT) algorithms $\text{DVRS} = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verify}, \text{Sim})$ as follows:

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: The algorithm initializes public parameters pp using λ as input, where $\lambda \in \mathbb{N}$ is the security parameter. pp is implicit for all algorithms below if not explicitly mentioned.
- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$: The algorithm generates and outputs a randomized key pair (pk, sk) from its input pp .
- $\Sigma \leftarrow \text{Sign}(\text{R}, \text{pk}_D, \text{sk}_\pi, M)$: The algorithm inputs the public key ring $\text{R} = \{\text{pk}_1, \dots, \text{pk}_N\}$ and designated verifier's public key pk_D , the signer's secret key sk_π , and a message M . It requires that the signer's public key $\text{pk}_\pi \in \text{R}$. Finally, it outputs a signature Σ for the message M .

- $\{0, 1\} \leftarrow \text{Verify}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M, \Sigma)$: The algorithm inputs the public key ring \mathbf{R} and designated verifier's key pair $(\mathbf{pk}_D, \mathbf{sk}_D)$, the message M and signature Σ . If (M, Σ) is valid, it outputs 1. Otherwise, outputs 0.
- $\Sigma' \leftarrow \text{Sim}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M)$: The algorithm inputs the public key ring and designated verifier's key pair $(\mathbf{pk}_D, \mathbf{sk}_D)$, and a message M . Finally, it outputs a simulated signature Σ' for the message M .

The strong DVRS scheme should satisfy correctness: any signature Σ generated by the honest signer and any simulated signature Σ' generated by the honest designated verifier should both successfully pass the `Verify`.

Definition 1 (Correctness). *The DVRS scheme DVRS satisfies correctness, if the following verification equations hold.*

$$\begin{aligned} \text{Verify}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M, \Sigma = \text{Sign}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_\pi, M)) &= 1. \\ \text{Verify}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M, \Sigma' = \text{Sim}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M)) &= 1. \end{aligned}$$

2.2 Security Definitions

We use the following oracles to capture the adversary \mathcal{A} 's capabilities.

1. **Corruption Oracle** $\mathbf{sk} \leftarrow \mathcal{O}_{\text{Cor}}(\mathbf{pk})$: On query a public key $\mathbf{pk} \in \mathcal{S}_{\text{pk}}$, it returns the secret key $\mathbf{sk} \in \mathcal{S}_{\text{sk}}$, such that $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}(\text{pp})$.
2. **Signing Oracle** $\Sigma \leftarrow \mathcal{O}_{\text{Sign}}(\mathbf{R}, \mathbf{pk}_D, \mathbf{pk}, M)$: On query the public key ring \mathbf{R} , the designated verifier's public key \mathbf{pk}_D , the signer's public key \mathbf{pk} , and a message M , it returns a valid signature Σ such that $\Sigma \leftarrow \text{Sign}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}, M)$, where $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}(\text{pp})$ and $\mathbf{pk} \in \mathbf{R}$.
3. **Verification Oracle** $\{0, 1\} \leftarrow \mathcal{O}_{\text{Verify}}(\mathbf{R}, \mathbf{pk}_D, M, \Sigma)$: On query the public key ring \mathbf{R} and designated verifier's public key \mathbf{pk}_D , the message M and signature Σ . If (M, Σ) is valid, it outputs 1. Otherwise, it outputs 0.
4. **Simulation Oracle** $\Sigma' \leftarrow \mathcal{O}_{\text{Sim}}(\mathbf{R}, \mathbf{pk}_D, M)$: On query the public key ring \mathbf{R} and designated verifier's public key \mathbf{pk}_D , and a message M , it returns a valid simulated signature Σ' such that $\Sigma' \leftarrow \text{Sim}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M)$.

Unforgeability. In the strong DVRS scheme, unforgeability implies the fact that any user whose public key does not belong to the public key ring \mathbf{R} , should be unable to produce a valid signature on behalf of \mathbf{R} . We inherit and modify the strongest definition of unforgeability w.r.t. insider corruption in [5], which stipulates that even if the adversary can adaptively corrupt some honest users and acquire their secret keys, it still cannot succeed in forging a new signature.

Definition 2 (Unforgeability w.r.t. insider corruption). *Given the strong DVRS scheme DVRS and a PPT adversary \mathcal{A} , consider the following game:*

- *Keypairs are generated by $\text{KeyGen}(\text{pp})$, and the set of public keys \mathbf{S} is given to \mathcal{A} , whose cardinality $|\mathbf{S}| = q_k$.*

- \mathcal{A} is given access to \mathcal{O}_{Cor} , which returns the set of corrupted public keys S_{Cor} .
- \mathcal{A} is given access to $\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}, \mathcal{O}_{\text{Verify}}$.
- \mathcal{A} outputs $(R^*, \text{pk}_D^*, M^*, \Sigma^*)$ to the challenger \mathcal{C} , and wins if
 1. $\text{Verify}(R^*, \text{pk}_D^*, \text{sk}_D^*, M^*, \Sigma^*) = 1$.
 2. $(R^*, \text{pk}_D^*, M^*, \dots)$ has not been queried to $\mathcal{O}_{\text{Sign}}$ and \mathcal{O}_{Sim} .
 3. $R^* \cup \{\text{pk}_D^*\} \subseteq S \setminus S_{\text{Cor}}$.

The advantage is $\text{Adv}_{\mathcal{A}}^{UF}(\lambda) = \left| \Pr[\mathcal{A} \text{ wins the game}] \right|$.

The DVRS scheme DVRS is unforgeability w.r.t. insider corruption, if for any PPT adversary \mathcal{A} , the above advantage $\text{Adv}_{\mathcal{A}}^{UF}(\lambda)$ is negligible.

Signer Anonymity. In the strong DVRS scheme, signer anonymity implies that any user (including the designated verifier) attempting to deduce the real signer's identity in the ring from a ring signature, would have a negligible advantage over random guessing among all the ring members. We inherit and modify the strongest definition of Signer Anonymity against full key exposure in [5], where the adversary \mathcal{A} is given all randomnesses to generate secret keys.

Definition 3 (Signer Anonymity against full key exposure). Given the strong DVRS scheme DVRS and an unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following game:

- Keypairs are generated by $\text{KeyGen}(\text{pp}; r_i)$ for randomly chosen r_i , and the set of public keys S is given to \mathcal{A} , whose cardinality $|S| = q_k$.
 - \mathcal{A} is given access to $\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}, \mathcal{O}_{\text{Verify}}$.
 - \mathcal{A}_1 chooses the public key ring $R^* \subseteq S$ and $\text{pk}_{i_0}^*, \text{pk}_{i_1}^* \in R^*$, the designated verifier's public key $\text{pk}_D^* \subseteq S$, a message $M^* \in \mathcal{M}$, and sends them to the challenger \mathcal{C} . \mathcal{C} chooses a random bit $b \leftarrow_{\$} \{0, 1\}$ and computes a challenge signature $\Sigma_b = \text{Sign}(R^*, \text{pk}_D^*, \text{sk}_{i_b}^*, M^*)$ and returns. Then, \mathcal{A}_2 , inheriting the state of \mathcal{A}_1 and r_j for generating R^* , attempts to guess b' , and wins if $b' = b$.
- The advantage is $\text{Adv}_{\mathcal{A}}^{SA}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$.

The DVRS scheme DVRS is perfect signer anonymity against full key exposure, if for any unbounded adversary \mathcal{A} , the above advantage $\text{Adv}_{\mathcal{A}}^{SA}(\lambda)$ is negligible.

Non-Transferability. In the strong DVRS scheme, Non-Transferability implies that only the designated verifier can validate the signature, and it cannot be used to convince others. We adopt the strongest definition to our knowledge, that is, all the secret keys are given to the adversary \mathcal{A} . In other words, when provided a signature produced by the Sign algorithm, the designated verifier can invoke the Sim algorithm to yield a simulated signature associated with it, which are statistical indistinguishable.

Definition 4 (Non-Transferability). Given the strong DVRS scheme DVRS and an unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following game:

- *Keypairs are generated by $\text{KeyGen}(\text{pp})$, and the set of public keys S is given to \mathcal{A} , whose cardinality $|S| = q_k$. All the secret keys are also obtained by \mathcal{A} .*
- *\mathcal{A}_1 chooses the public key ring $R^* \subseteq S$ and $\text{pk}_\pi^* \in R^*$, the designated verifier's public key $\text{pk}_D^* \subseteq S$, a message $M^* \in \mathcal{M}$, and sends them to the challenger \mathcal{C} . \mathcal{C} chooses a random bit $b \leftarrow_{\$} \{0, 1\}$ to compute a challenge signature Σ_b and returns, where $\Sigma_0 = \text{Sign}(R^*, \text{pk}_D^*, \text{sk}_\pi^*, M^*)$ and $\Sigma_1 = \text{Sim}(R^*, \text{pk}_D^*, \text{sk}_D^*, M^*)$. Then, \mathcal{A}_2 , inheriting the state of \mathcal{A}_1 , attempts to guess b' , and wins if $b' = b$. The advantage is $\text{Adv}_{\mathcal{A}}^{NT}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$.*

The DVRS scheme DVRS is perfect non-transferability, if for any unbounded adversary \mathcal{A} , the above advantage $\text{Adv}_{\mathcal{A}}^{NT}(\lambda)$ is negligible.

3 Constructions

In this section, we present TripleRing, a generic construction for strong DVRS. We begin with our starting point, Extended Type-T* Canonical Identification, which is also newly defined and can be instantiated from various assumptions.

3.1 Type-T Canonical Identifications and Extensions

To describe generic conversion from identifications to ring signature, the Type-T Canonical Identification was extracted [1, 28]. It proceeds as below.

Let \mathcal{S}_c and \mathcal{S}_y denote the spaces for the challenge c and the randomness y . The definitions for other randomness spaces follow similarly. Moreover, A, Z, V denote commit function, response function, and verification function respectively.

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , the algorithm defines public parameters pp and outputs them. pp is implicit for all algorithms below if not explicitly mentioned.
- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$: The algorithm generates and outputs a randomized key pair (pk, sk) from its input pp .
- $(y, Y) \leftarrow \text{Proof}_1(\text{sk})$: The algorithm inputs the secret key sk and chooses $y \leftarrow_{\$} \mathcal{S}_y$. Then, it runs the commit function $Y = A(y)$, and outputs (y, Y) , where the commitment Y can be public to all.
- $c \leftarrow \text{Ch}(Y)$: The algorithm inputs the commitment Y , and outputs a random $c \leftarrow_{\$} \mathcal{S}_c$ as the challenge.
- $z \leftarrow \text{Proof}_2(\text{sk}, y, c)$: The algorithm takes the secret key sk , the input y for the function A in Proof_1 , and the challenge c from ch as inputs. It runs the response function $z = Z(\text{sk}, y, c)$ to generate and output a response z .
- $\{0, 1\} \leftarrow \text{Verify}(\text{pk}, z, c)$: The algorithm inputs the public key pk , the response z and challenge c , runs the verification function $Y' = V(\text{pk}, z, c)$. It checks $c \stackrel{?}{=} \text{Ch}(Y')$. If it holds, returns 1, otherwise returns 0.

Recently, a new Type-T* Canonical Identification was defined [28], with the following operations and properties added.

1. The challenge space \mathcal{S}_c is a commutative group with operation \otimes , while \oslash is its inverse operation in this group.
2. If c_1 and c_2 are uniformly distributed in \mathcal{S}_c , then $c_1 \otimes c_2$ is also uniformly distributed in \mathcal{S}_c .
3. The verification function $V(\text{pk}, z, c)$ can be split into $V_1(z) \odot V_2(\text{pk}, c)$, where \odot is a commutative group operation for the range of V .
4. The verification function $V_1(z)$ has additive/multiplicative homomorphism, i.e., $V_1(z_1) \odot V_1(z_2) = V_1(z_1 \oplus z_2)$, where \oplus is a commutative group operation in the range of z , and the homomorphic operation is efficiently computed.
5. Given the secret key sk matched with pk and c , there exists a function \mathcal{I}_V can be efficiently computed such that $V_1(\mathcal{I}_V(\text{sk}, c)) = V_2(\text{pk}, c)$.

We define an additional simulation function $S(\text{pk}_D, x, w)$ that satisfies the following properties, leading to the Extended Type-T* Canonical Identification.

1. The space \mathcal{S}_w is a commutative group with operation \boxtimes , while \boxdiv is its inverse operation in this group.
2. If w_1 and w_2 are uniformly distributed in \mathcal{S}_w , then $w_1 \boxtimes w_2$ is also uniformly distributed in \mathcal{S}_w .
3. The simulation function $S(\text{pk}_D, x, w)$ can be split into $S_1(x) \boxtimes S_2(\text{pk}_D, w)$, where \boxtimes is a commutative group operation for the range of S .
4. The simulation function $S_1(x)$ has additive/multiplicative homomorphism, i.e., $S_1(x_1) \boxtimes S_1(x_2) = S_1(x_1 \boxplus x_2)$, where \boxplus is a commutative group operation for the range of x , and the homomorphic operation is efficiently computed.
5. Given the secret key sk_D matched with pk_D and w , there exists a function \mathcal{I}_S can be efficiently computed such that $S_1(\mathcal{I}_S(\text{sk}_D, w)) = S_2(\text{pk}_D, w)$.

In particular, we have the special case of the Extended Type-T* Canonical Identification that simultaneously satisfy the following properties.

- $\mathcal{S}_w = \mathcal{S}_c$, then the operations \otimes and \oslash in \mathcal{S}_c are applicable to \mathcal{S}_w , allowing \boxtimes and \boxdiv to be replaced with \otimes and \oslash .
- The range of x is the same as that of z , then the operation \oplus in the former are applicable to the latter, allowing \boxplus to be replaced with \oplus .
- The range of the simulation function S is the same as that of the verification function V , then the operation \odot in the former are applicable to the latter, allowing \boxtimes to be replaced with \odot .

We follow the *Special Impersonation under Key Only Attack* for our Extended Type-T* Canonical Identification. It combines aspects of special soundness and impersonation attacks, effectively addressing the knowledge gap in lattice-based zero-knowledge proofs [17, 27]. For more details, please refer to [28].

Definition 5 (Special Impersonation under Key Only Attack, [28]). A Canonical Identification $\{\text{Setup}, \text{KeyGen}, \text{Proof}_1, \text{Ch}, \text{Proof}_2, \text{Verify}\}$ is special impersonation under key only attack, if the advantage for a PPT adversary \mathcal{A} to output two valid transcripts for the same commitment is negligible, i.e.,

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda) \\ (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp}) \\ z, c, z', c' \leftarrow \mathcal{A}(\text{pp}, \text{pk}) \end{array} : \begin{array}{l} \text{Verify}(\text{pk}, z, c) = 1 \\ \text{Verify}(\text{pk}, z', c') = 1 \\ c \neq c' \wedge c, c' \in \mathcal{S}_c \end{array} \right] \leq \text{negl}(\lambda).$$

3.2 From Extended Type-T* Canonical Identification to TripleRing

We first give some intuition for construction of Type-T DVRS. To make it more intuitive, we use an example of the Schnorr scheme [23] running in a group \mathbb{G} denoted inside $\langle \cdot \rangle$, whose secret key is sk and public key is $\text{pk} = g^{\text{sk}}$, and a message M to be signed. Here, $\leftarrow_{\$}$ denotes sampling uniformly at random.

To be more precise, as illustrated in **Fig. 1**, a signer possessing the secret key sk_{π} (associated with a public key $\text{pk}_{\pi} \in \mathcal{R} = \{\text{pk}_1, \dots, \text{pk}_N\}$) generates a signature using the Type-T DVRS.Sign Algorithm. Meanwhile, a designated verifier with the secret key sk_D (associated with the public key pk_D) generates a simulated signature using the Type-T DVRS.Sim Algorithm.

| Type-T DVRS.Sign(pp, R, pk_D , sk_{π} , M) | Type-T DVRS.Sim(pp, R, pk_D , sk_D , M) |
|--|---|
| for $j = \pi$ do sample $y_{\pi} \leftarrow_{\$} \mathcal{S}_y$, $w_{\pi} \leftarrow_{\$} \mathcal{S}_c$, $x_{\pi} \leftarrow_{\$} \mathcal{S}_x$, $\langle y_{\pi} \leftarrow_{\$} \mathcal{Z}_q, w_{\pi} \leftarrow_{\$} \mathcal{Z}_q, x_{\pi} \leftarrow_{\$} \mathcal{Z}_q \rangle$ $Y_{\pi} = A(y_{\pi})$, $W_{\pi} = S(\text{pk}_D, x_{\pi}, w_{\pi})$, $\langle Y_{\pi} = g^{y_{\pi}}, W_{\pi} = g^{x_{\pi}} \cdot \text{pk}_D^{w_{\pi}} \rangle$ $c_{\pi+1} = H(\mathcal{R}, \text{pk}_D, M, Y_{\pi}, W_{\pi})$; endfor for $j = \pi + 1, \dots, N - 1, N, 1, \dots, \pi - 1$ do sample $z_j \leftarrow_{\$} \mathcal{S}_z$, $w_j \leftarrow_{\$} \mathcal{S}_c$, $x_j \leftarrow_{\$} \mathcal{S}_x$, $\langle z_j \leftarrow_{\$} \mathcal{Z}_q, w_j \leftarrow_{\$} \mathcal{Z}_q, x_j \leftarrow_{\$} \mathcal{Z}_q \rangle$ $Y_j = V(\text{pk}_j, z_j, c_j \otimes w_j)$, $W_j = S(\text{pk}_D, x_j, w_j)$, $\langle Y_j = g^{z_j} \cdot \text{pk}_j^{c_j + w_j}, W_j = g^{x_j} \cdot \text{pk}_D^{w_j} \rangle$ $c_{(j \bmod N)+1} = H(\mathcal{R}, \text{pk}_D, M, Y_j, W_j)$; endfor // Compute the missing z_{π} $z_{\pi} = Z(\text{sk}_{\pi}, y_{\pi}, c_{\pi} \otimes w_{\pi})$; $\langle z_{\pi} = y_{\pi} - (c_{\pi} + w_{\pi}) \cdot \text{sk}_{\pi} \rangle$ return $\Sigma = (c_1, \{z_j\}_{j=1}^N, \{x_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$ | for $j = 1$ do sample $z'_1 \leftarrow_{\$} \mathcal{S}_z$, $\eta \leftarrow_{\$} \mathcal{S}_c$, $\phi \leftarrow_{\$} \mathcal{S}_y$, $\langle z'_1 \leftarrow_{\$} \mathcal{Z}_q, \eta \leftarrow_{\$} \mathcal{Z}_q, \phi \leftarrow_{\$} \mathcal{Z}_q \rangle$ $Y'_1 = V(\text{pk}_1, z'_1, \eta)$, $W'_1 = A(\phi)$, $\langle Y'_1 = g^{z'_1} \cdot \text{pk}_1^{\eta}, W'_1 = g^{\phi} \rangle$ $c'_2 = H(\mathcal{R}, \text{pk}_D, M, Y'_1, W'_1)$; endfor for $j = 2, \dots, N$ do sample $z'_j \leftarrow_{\$} \mathcal{S}_z$, $w'_j \leftarrow_{\$} \mathcal{S}_c$, $x'_j \leftarrow_{\$} \mathcal{S}_x$, $\langle z'_j \leftarrow_{\$} \mathcal{Z}_q, w'_j \leftarrow_{\$} \mathcal{Z}_q, x'_j \leftarrow_{\$} \mathcal{Z}_q \rangle$ $Y'_j = V(\text{pk}_j, z'_j, c'_j \otimes w'_j)$, $W'_j = S(\text{pk}_D, x'_j, w'_j)$, $\langle Y'_j = g^{z'_j} \cdot \text{pk}_j^{c'_j + w'_j}, W'_j = g^{x'_j} \cdot \text{pk}_D^{w'_j} \rangle$ $c'_{(j \bmod N)+1} = H(\mathcal{R}, \text{pk}_D, M, Y'_j, W'_j)$; endfor // Compute the missing w'_1 and x'_1 $w'_1 = \eta \odot c'_1$, $x'_1 = Z(\text{sk}_D, \phi, w'_1)$; $\langle w'_1 = \eta - c'_1, x'_1 = \phi - w'_1 \cdot \text{sk}_D \rangle$ return $\Sigma = (c'_1, \{z'_j\}_{j=1}^N, \{x'_j\}_{j=1}^N, \{w'_j\}_{j=1}^N)$ |

Fig. 1. Type-T DVRS.Sign Algorithm and Type-T DVRS.Sim Algorithm

To verify a signature $\Sigma = (c_1, \{z_j\}_{j=1}^N, \{x_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$, for $j = 1, \dots, N$, it sequentially reconstructs $Y_j = V(\text{pk}_j, z_j, c_j \otimes w_j)$, $W_j = S(\text{pk}_D, x_j, w_j)$, and the challenge $c_{j+1} = H(\mathcal{R}, \text{pk}_D, M, Y_j, W_j)$. Finally, it checks correctness of $c_1 \stackrel{?}{=} H(\mathcal{R}, \text{pk}_D, M, Y_N, W_N)$, and outputs 1 (valid)/ 0 (invalid). The verification for a simulated signature is in the same way. According to the generation of z_{π} in the Type-T DVRS.Sign Algorithm (w'_{π} and x'_{π} in the Type-T DVRS.Sim Algorithm), it is not hard to know that the verification is established.

We further develop Type-T DVRS to TripleRing. Roughly, we separate the ring formed in Type-T DVRS Sign (Sim) Algorithm into a Challenge-Ring and

a Response-Ring (Simulation-Ring). The underlying idea is motivated by [28]: using the commutative group operation \odot (e.g., modular addition and modular multiplication) to separate the verification function $V = V(\mathbf{pk}, z, c) = V_1(z) \odot V_2(\mathbf{pk}, c)$ ($V_1 : g^z, V_2 : \mathbf{pk}^c$, and \odot indicates \cdot in the group \mathbb{G}) and simulation function $S = S(\mathbf{pk}_D, x, w) = S_1(x) \odot S_2(\mathbf{pk}_D, w)$ ($S_1 : g^x, S_2 : \mathbf{pk}_D^w$).

As depicted in **Fig. 2**, the signer forms the Response-Ring and connects it with the Challenge-Ring (during the signing), whereas the designated verifier forms the Response-Ring and connects it with the Simulation-Ring (during the simulation). The hidden/recovery functions that are compatible were also integrated, keeps the total signature sizes small while enhancing the designated verifier property from weak to strong. The naming of TripleRing also originates from the formation of three interconnected rings.

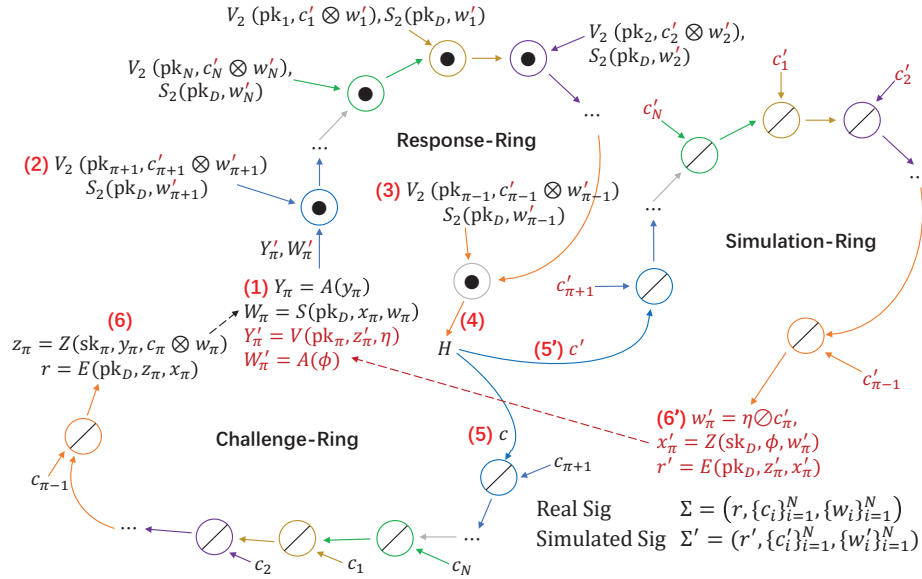


Fig. 2. Our TripleRing generic construction for strong DVRS

To be more specific, the signer first picks $y_\pi \leftarrow_{\$} \mathcal{S}_y$ and $\{c_j\}_{j \neq \pi}, \{w_j\}_{j=1}^N$, and $\{x_j\}_{j=1}^N$ from their respective ranges. Then, it forms a Response-Ring via the group operation \odot . We first add up all V_2 according to commutative:

$$Y_\pi = A(y_\pi) \odot \bigodot_{j \neq \pi} V_2(\mathbf{pk}_j, c_j \otimes w_j), \quad W_\pi = S(\mathbf{pk}_D, x_\pi, w_\pi) \odot \bigodot_{j \neq \pi} S_2(\mathbf{pk}_D, w_j). \quad (1)$$

Then, it forms a Challenge-Ring by computing $c = H(R, \mathbf{pk}_D, M, Y_\pi, W_\pi)$ and the missing challenge $c_\pi = c \odot (\bigotimes_{j \neq \pi} c_j)$. The following equation should hold:

$$c_1 \otimes c_2 \otimes \cdots \otimes c_N = c = H(R, \mathbf{pk}_D, M, Y_\pi, W_\pi). \quad (2)$$

Finally, the Challenge-Ring is connected via the computation of the challenge $z_\pi = Z(\mathbf{sk}_\pi, y_\pi, c_\pi \otimes w_\pi)$. After adding a hidden layer to derive r from (z_π, x_π) , the signature is $\Sigma = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$. During the simulation, the designated verifier, who holds the secret key \mathbf{sk}_D , can run a similar procedure to the above to generate a simulated signature $\Sigma' = (r', \{c'_j\}_{j=1}^N, \{w'_j\}_{j=1}^N)$. The simulated signature Σ' can be proven indistinguishable from the real signature Σ .

During the verification of Σ , the designated verifier first recover (z_π, x_π) . Then, using underlying relations $V(\mathbf{pk}_\pi, z_\pi, c_\pi \otimes w_\pi) = V_1(z_\pi) \odot V_2(\mathbf{pk}_\pi, c_\pi \otimes w_\pi)$ and $S(\mathbf{pk}_D, x_\pi, w_\pi) = S_1(x_\pi) \odot S_2(\mathbf{pk}_D, w_\pi)$, it can reconstruct (1) as

$$Y_\pi = V_1(z_\pi) \odot \bigodot_{j=1}^N V_2(\mathbf{pk}_j, c_j \otimes w_j), \quad W_\pi = S_1(x_\pi) \odot \bigodot_{j=1}^N S_2(\mathbf{pk}_D, w_j). \quad (3)$$

Finally, it checks whether $H(\mathbf{R}, \mathbf{pk}_D, M, Y_\pi, W_\pi) \stackrel{?}{=} \bigotimes_{j=1}^N c_j$ to verify (2), which represents the legality of Σ . The verification of the simulated Σ' is similar.

3.3 TripleRing: A Generic Construction for strong DVRS

TripleRing works as below, following the notations consistent with Section 3.2.

- $\mathbf{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , this algorithm defines the message space \mathcal{M} , the secret key space $\mathcal{S}_{\mathbf{sk}}$ and public key space $\mathcal{S}_{\mathbf{pk}}$, the space \mathcal{S}_y and challenge space \mathcal{S}_c (we let $\mathcal{S}_w = \mathcal{S}_c$ as in the special case in Section 3.1). It also defines the hash function $H : \{0, 1\}^* \rightarrow \mathcal{S}_c$, commit function A , response function Z , verification function $V = V_1 \odot V_2$, simulation function $S = S_1 \odot S_2$, and hidden function E and recovery function F . Finally, it outputs them as public parameters \mathbf{pp} . \mathbf{pp} is implicit for all algorithms below if not explicitly mentioned.
- $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}(\mathbf{pp})$: The algorithm generates and outputs a randomized key pair $(\mathbf{pk}, \mathbf{sk})$ from its input \mathbf{pp} .
- $\Sigma \leftarrow \text{Sign}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_\pi, M)$: On input $\{\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_\pi, M\}$, it runs as below.
 1. Chooses $y_\pi \leftarrow_{\$} \mathcal{S}_y$ and $\{c_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c, \{w_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, \{x_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_y$;
 2. Computes

$$Y_\pi = A(y_\pi) \odot \bigodot_{j \neq \pi} V_2(\mathbf{pk}_j, c_j \otimes w_j), \quad W_\pi = S(\mathbf{pk}_D, x_\pi, w_\pi) \odot \bigodot_{j \neq \pi} S_2(\mathbf{pk}_D, w_j),$$

$$c = H(\mathbf{R}, \mathbf{pk}_D, M, Y_\pi, W_\pi), \quad c_\pi = c \oslash \left(\bigotimes_{j \neq \pi} c_j \right), \quad z_\pi = Z(\mathbf{sk}_\pi, y_\pi, c_\pi \otimes w_\pi);$$

3. Computes $r = E(\mathbf{pk}_D, z_\pi, x_\pi)$ according to the hidden function E .
- The final signature for M is $\Sigma = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$.
- $\{0, 1\} \leftarrow \text{Verify}(\mathbf{R}, \mathbf{pk}_D, \mathbf{sk}_D, M, \Sigma)$: On input $\{\mathbf{R}, \mathbf{pk}_D, M, \Sigma\}$, the algorithm verifies the signature Σ as follows.
 1. Parses $\Sigma = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$, and uses the recovery function $F(\mathbf{sk}_D, r)$ to compute $V_1(z_\pi)$ and $S_1(x_\pi)$;

2. Computes $c = H(R, \text{pk}_D, M, Y_\pi, W_\pi)$, where

$$Y_\pi = V_1(z_\pi) \odot \bigodot_{j=1}^N V_2(\text{pk}_j, c_j \otimes w_j), \quad W_\pi = S_1(x_\pi) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j);$$

3. Checks whether $c \stackrel{?}{=} \bigotimes_{j=1}^N c_j$, and returns 1 for valid/ 0 for invalid.

Remarks: Extra checking may be necessary, e.g., rejection samplings [18].

– $\Sigma' \leftarrow \text{Sim}(R, \text{pk}_D, \text{sk}_D, M)$: On input $\{R, \text{pk}_D, \text{sk}_D, M\}$, it runs as below.

1. Chooses $\eta, \{c'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c, \{w'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c$, and $z'_\pi, \phi, \{x'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_y$;
2. Computes

$$Y'_\pi = V(\text{pk}_\pi, z'_\pi, \eta) \odot \bigodot_{j \neq \pi} V_2(\text{pk}_j, c'_j \otimes w'_j), \quad W'_\pi = A(\phi) \odot \bigodot_{j \neq \pi} S_2(\text{pk}_D, w'_j),$$

$$c' = H(R, \text{pk}_D, M, Y'_\pi, W'_\pi), \quad c'_\pi = c' \oslash \left(\bigotimes_{j \neq \pi} c'_j \right),$$

$$w'_\pi = \eta \oslash c'_\pi, \quad x'_\pi = Z(\text{sk}_D, \phi, w'_\pi);$$

3. Computes $r' = E(\text{pk}_D, z'_\pi, x'_\pi)$ according to the hidden function E .

The final simulated signature for M is $\Sigma' = (r', \{c'_j\}_{j=1}^N, \{w'_j\}_{j=1}^N)$.

Theorem 1 (Correctness). *TripleRing satisfies correctness in Definition 1.*

Proof. We defer it to Appendix A. \square

Theorem 2 (Unforgeability). *TripleRing has unforgeability w.r.t. insider corruption in the random oracle model, if the underlying Extended Type- T^* Canonical Identification is secure against special impersonation under key only attack, and the cardinality of the challenge space $|\mathcal{S}_c| > \max\{q_{\text{sig}}(q_h + q_{\text{sig}} - 1), q_{\text{sim}}(q_h + q_{\text{sim}} - 1)\}$, where $q_{\text{sig}}, q_{\text{sim}}$ and q_h are the number of queries to Signing Oracle $\mathcal{O}_{\text{Sign}}$, Simulation Oracle \mathcal{O}_{Sim} and Random Oracle \mathcal{H} , respectively.*

Proof. We defer it to Appendix B. \square

Theorem 3 (Signer Anonymity). *TripleRing has perfect signer anonymity against full key exposure in the random oracle model, if the cardinality of the challenge space $|\mathcal{S}_c| > \max\{q_{\text{sig}}(q_h + q_{\text{sig}} - 1), q_{\text{sim}}(q_h + q_{\text{sim}} - 1)\}$, where $q_{\text{sig}}, q_{\text{sim}}$ and q_h are the number of queries to Signing Oracle $\mathcal{O}_{\text{Sign}}$, Simulation Oracle \mathcal{O}_{Sim} and Random Oracle \mathcal{H} , respectively.*

Proof. We defer it to Appendix C. \square

Theorem 4 (Non-Transferability). *TripleRing has perfect non-transferability in the random oracle model, if the cardinality of the challenge space $|\mathcal{S}_c| > \max\{q_{\text{sig}}(q_h + q_{\text{sig}} - 1), q_{\text{sim}}(q_h + q_{\text{sim}} - 1)\}$, where $q_{\text{sig}}, q_{\text{sim}}$ and q_h are the number of queries to Signing Oracle $\mathcal{O}_{\text{Sign}}$, Simulation Oracle \mathcal{O}_{Sim} and Random Oracle \mathcal{H} , respectively.*

Proof. We defer it to Appendix D. \square

Comparison with other DVRS Schemes. The existing DVRS schemes with rigorous provable security [3,4] both can be viewed as DL-based instances of our Type-T Construction in Section 3.2. According to our notations, they adopt the simulation function that differs from that of our DL-based instance in $\langle \cdot \rangle$. However, the simulation function they selected makes it difficult to generalize to assumptions other than DL. We not only achieve the stronger designated verifier property in our generic construction, but also provide instances which is logarithmic-sizes/conjectured quantum-resistant in the next section.

4 Instantiations

4.1 TripleRing-EC: A Logarithmic-Size Instance from DL and DDH

Notations and Assumptions. We denote by \mathbb{G} a cyclic group of order prime p . The vector $\mathbf{a} = [a_1, \dots, a_n]$ represents a vector comprising $a_i \in \mathbb{Z}_p$, while $\mathbf{a}_{[1:N']}$ and $\mathbf{a}_{[N'+1:n]}$ represent $[a_1, \dots, a_{N'}]$ and $[a_{N'+1}, \dots, a_n]$. The vector $\mathbf{g} = [g_1, \dots, g_n]$ denotes a vector comprising $g_i \in \mathbb{G}$. The notations $\mathbf{a} + b$ and $\mathbf{a} + \mathbf{b}$ represent $[a_1 + b, \dots, a_n + b]$ and $[a_1 + b_1, \dots, a_n + b_n]$, respectively. The notations $\langle \mathbf{a}, \mathbf{b} \rangle$ represents the inner product $\sum_{i=1}^n a_i b_i$, and $\mathbf{a} \circ \mathbf{b}$ denotes the Hadamard product $[a_1 b_1, \dots, a_n b_n]$. $\mathbf{g}^{\mathbf{a}}$ represents $[g_1^{\mathbf{a}}, \dots, g_n^{\mathbf{a}}]$, $g^{\mathbf{a}}$ denotes $[g^{a_1}, \dots, g^{a_n}]$, and $\mathbf{g}^{\mathbf{a}}$ denotes $[g_1^{a_1}, \dots, g_n^{a_n}]$. We use $\leftarrow_{\$}$ denotes uniform random sampling.

Definition 6 (Discrete Logarithm, [23]). *Given a cycle group \mathbb{G} of prime order p , and two group elements $g, h \leftarrow_{\$} \mathbb{G}$ that are not the identity element of \mathbb{G} , find s such that $h = g^s$.*

We now present the settings for a DL-based Extended Type-T* Canonical Identification, which can be converted into the strong DVRS using TripleRing.

- The public parameters include the cycle group \mathbb{G} of the prime order p . The space $\mathcal{S}_y = \mathbb{Z}_p$ has a communicative operation $\oplus = +$ in it, and the challenge space $\mathcal{S}_c = \mathbb{Z}_p$ has communicative operations $\otimes = +$ and $\oslash = -$ in it.
- For user i , the secret key is $\mathbf{sk}_i \leftarrow_{\$} \mathbb{Z}_p$, and public key is $\mathbf{pk}_i = g^{\mathbf{sk}_i} \in \mathbb{G}$;
- For the designated verifier, the secret key is $\mathbf{sk}_D \leftarrow_{\$} \mathbb{Z}_p$, and public key is $\mathbf{pk}_D = g^{\mathbf{sk}_D} \in \mathbb{G}$;
- The commit function $A(y) := g^y$ for $y \leftarrow_{\$} \mathcal{S}_y = \mathbb{Z}_p$, in $\text{Proof}_1(\mathbf{sk})$;
- The hash function $H : \{0, 1\}^* \rightarrow c \in \mathcal{S}_c = \mathbb{Z}_p$;
- The response function $Z(\mathbf{sk}, y, c) := y - c \cdot \mathbf{sk}$, and we have $z = y - c \cdot \mathbf{sk}$;
- The verification function $V(\mathbf{pk}, z, c) = V_1(z) \cdot V_2(\mathbf{pk}, c) = g^z \cdot \mathbf{pk}^c$;
- The simulation function $S(\mathbf{pk}_D, x, w) = S_1(x) \cdot S_2(\mathbf{pk}_D, w) = g^x \cdot \mathbf{pk}_D^w$;
- The hidden function E and recovery function F . Since $z_\pi, x_\pi \in \mathcal{S}_y = \mathbb{Z}_p$, we have both $V_1(z_\pi) = g^{z_\pi}$ and $S_1(x_\pi) = g^{x_\pi}$ in the group \mathbb{G} . We set $r = (r_1, r_2, r_3, r_4) = E(\mathbf{pk}_D, z_\pi, x_\pi) = (g^{u_z}, \mathbf{pk}_D^{u_z} \cdot g^{z_\pi}, g^{u_x}, \mathbf{pk}_D^{u_x} \cdot g^{x_\pi}) \in \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$, where $u_z, u_x \leftarrow_{\$} \mathbb{Z}_p$. Then, we have $F(\mathbf{sk}_D, r) = (r_2/r_1^{\mathbf{sk}_D}, r_4/r_3^{\mathbf{sk}_D}) = (V_1(z_\pi), S_1(x_\pi))$. Correctness and security are essentially equivalent to those of the ElGamal PKE [10], which rely on DDH Assumptions.

We observe that the settings satisfy the following properties in Section 3.1.

1. The challenge space $\mathcal{S}_c = \mathbb{Z}_p$ is a commutative group, with operation $\otimes = +$, while $\oslash = -$ is its inverse operation;
2. If c_1 and c_2 are uniformly distributed in \mathcal{S}_c , then $c_1 \otimes c_2$ is also uniformly distributed in \mathcal{S}_c . This also holds true for $w_1 \otimes w_2$ and $c_1 \otimes w_1$;
3. The verification function $V(\mathbf{pk}, z, c) = g^z \cdot \mathbf{pk}^c = V_1(z) \cdot V_2(\mathbf{pk}, c)$ and simulation function $S(\mathbf{pk}_D, x, w) = g^x \cdot \mathbf{pk}_D^w = S_1(x) \cdot S_2(\mathbf{pk}_D, w)$, where $\odot = \cdot$ is a commutative group operation for their ranges;
4. The verification function $V_1(z) = g^z$ and simulation function $S_1(x) = g^x$ have additive homomorphism, i.e., $V_1(z_1) \odot V_1(z_2) = g^{z_1} \cdot g^{z_2} = g^{z_1+z_2} = V_1(z_1 \oplus z_2)$ and $S_1(x_1) \odot S_1(x_2) = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} = S_1(x_1 \oplus x_2)$, where $\oplus = +$ is a commutative group operation. Besides, the homomorphic operation can be efficiently computed;
5. Given \mathbf{sk} matched with \mathbf{pk} and c , there exists $\mathcal{I}_V(\mathbf{sk}, c) = c \cdot \mathbf{sk}$ such that $V_1(\mathcal{I}_V(\mathbf{sk}, c)) = g^{c \cdot \mathbf{sk}} = \mathbf{pk}^c = V_2(\mathbf{pk}, c)$. Meanwhile, given \mathbf{sk}_D matched with \mathbf{pk}_D and x , there exists $\mathcal{I}_S(\mathbf{sk}_D, w) = w \cdot \mathbf{sk}_D$ such that $S_1(\mathcal{I}_S(\mathbf{sk}_D, w)) = g^{w \cdot \mathbf{sk}_D} = \mathbf{pk}_D^w = S_2(\mathbf{pk}_D, w)$.

Theorem 5. *The above settings constitute an Extended Type- T^* Canonical Identification, which satisfies secure against special impersonation under key only attack in Definition 5, assuming the hardness of the Discrete Logarithm problem.*

Proof. We defer it to Appendix E. \square

According to TripleRing generic construction, the above settings can be adapted into a strong DVRS scheme TripleRing-DL, ensuring *unforgeability*, *signer anonymity*, and *non-transferability*.

Minus Argument. In order to decrease signature sizes into logarithmic, we devise Minus Argument. Then, we embed it into the above TripleRing-DL to obtain TripleRing-EC. Prior to it, we revisit Inner Product (IP) Argument in Bulletproofs [7]. Our Minus Argument can be seen as a generalization of it. Considering the following relations. Here, we denote $\{(\text{Public Input}; \text{Witness}) : \text{Relation}\}$ by the mathematical relation between Public Input and Witness.

$$\mathcal{R}_{\text{IP}} = \left\{ ((\mathbf{g}, \mathbf{h}, P, c); (\alpha, \beta)) \in (\mathbb{G}^N \times \mathbb{G}^N \times \mathbb{G} \times \mathbb{Z}_p) \times (\mathbb{Z}_p^N \times \mathbb{Z}_p^N) : \begin{array}{l} P = \prod_{j=1}^N g_j^{\alpha_j} h_j^{\beta_j} \wedge \gamma = \sum_{j=1}^N \alpha_j \beta_j \end{array} \right\};$$

$$\mathcal{R}_{\text{Sum}} = \{ ((\mathbf{g}, P, c); \alpha) \in (\mathbb{G}^N \times \mathbb{G} \times \mathbb{Z}_p) \times \mathbb{Z}_p^N : P = \prod_{j=1}^N g_j^{\alpha_j} \wedge \gamma = \sum_{j=1}^N \alpha_j \}.$$

Specifically, given $(\mathbf{g}, \mathbf{h}, P, c)$, a prover can use IP Argument to convince a verifier that it has the knowledge of $\alpha = [\alpha_1, \dots, \alpha_n] \in \mathbb{Z}_p^N$ and $\beta = [\beta_1, \dots, \beta_n] \in \mathbb{Z}_p^N$, such that $P = \prod_{j=1}^N g_j^{\alpha_j} h_j^{\beta_j}$ and $c = \sum_{j=1}^N \alpha_j \beta_j$. A straightforward method to prove \mathcal{R}_{Sum} in zero-knowledge is setting $\mathbf{h} = 1^N$ in IP Argument, while a more efficient Sum Argument was given in [28]. Both of them use the trick of recursion, which we generalize to prove the following relation $\mathcal{R}_{\text{Minus}}$ in **Algorithm 1**.

$$\mathcal{R}_{\text{Minus}} = \left\{ ((\mathbf{g}, g_D, P, Q, \Delta); (\alpha, \delta)) \in (\mathbb{G}^N \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p) \times (\mathbb{Z}_p^N \times \mathbb{Z}_p^N) : \right. \\ \left. P = \prod_{j=1}^N g_j^{\alpha_j} \quad \wedge \quad Q = g_D^{\delta_1 + \dots + \delta_n} \quad \wedge \quad \Delta = \sum_{j=1}^N \alpha_j - \sum_{j=1}^N \delta_j \right\}. \quad (4)$$

Algorithm 1 Non-Interactive Minus Argument (NIMA)

```

1: procedure NIMA.PROOF( $\{\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta\}, \alpha, \delta$ )
2:    $\Pi = (\mathbf{L}, \mathbf{R}, \alpha, \beta, \delta, \eta) \leftarrow \text{PF}(\mathbf{g}, g_D, u^{H'(P, u, \gamma)}, v^{H'(Q, v, \gamma - \Delta)}, \alpha, 1^N, \delta, 1^N)$ 
3: end procedure
4: procedure PF( $\mathbf{g}, g_D, \hat{u}, \hat{v}, \alpha, \beta, \delta, \eta$ )
5:   if  $N = 1$  then
6:     return  $\Pi = (\mathbf{L}, \mathbf{R}, \alpha, \beta, \delta, \eta)$ .
7:   else
8:     Compute  $N' = \frac{N}{2}$ , and add  $L_1, L_2, R_1, R_2 \in \mathbb{G}$  into initially empty lists
     L, R:
9:      $\gamma_{L_1} = \langle \alpha_{[N']}, \beta_{[N']} \rangle, \quad \gamma_{R_1} = \langle \alpha_{[N']}, \beta_{[N']} \rangle,$ 
10:     $L_1 = \mathbf{g}_{[N']}^{\alpha_{[N']}} \hat{u}^{\gamma_{L_1}}, \quad R_1 = \mathbf{g}_{[N']}^{\alpha_{[N']}} \hat{u}^{\gamma_{R_1}},$ 
11:     $\gamma_{L_2} = \langle \delta_{[N']}, \eta_{[N']} \rangle, \quad \gamma_{R_2} = \langle \delta_{[N']}, \eta_{[N']} \rangle,$ 
12:     $L_2 = g_D^{\delta_{[N']}} \hat{v}^{\gamma_{L_2}}, \quad R_2 = g_D^{\delta_{[N']}} \hat{v}^{\gamma_{R_2}};$ 
13:    Compute  $x = H(L_1, R_1)$  and  $y = H(L_2, R_2)$ ;
14:    Compute  $\mathbf{g}' = \mathbf{g}_{[N']}^{x^{-1}} \circ \mathbf{g}_{[N']}^x \in \mathbb{G}^{N'}, g'_D = g_D^{y+y^{-1}} \in \mathbb{G}$  and  $\alpha', \beta', \delta', \eta' \in \mathbb{Z}_p^{N'}$ :
15:     $\alpha' = x \cdot \alpha_{[N']} + x^{-1} \cdot \alpha_{[N']}, \quad \beta' = x^{-1} \cdot \beta_{[N']} + x \cdot \beta_{[N']},$ 
16:     $\delta' = y \cdot \delta_{[N']} + y^{-1} \cdot \delta_{[N']}, \quad \eta' = y^{-1} \cdot \eta_{[N']} + y \cdot \eta_{[N']};$ 
17:    return PF( $\mathbf{g}', g'_D, \hat{u}, \hat{v}, \alpha', \beta', \delta', \eta'$ );
18:   end if
19: end procedure
20: procedure NIMA.VERIFY( $\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta, \Pi = (\mathbf{L}, \mathbf{R}, \alpha, \beta, \delta, \eta)$ )
21:   Compute  $P' = P \cdot u^{\gamma \cdot H'(P, u, \gamma)}, \quad Q' = Q \cdot v^{(\gamma - \Delta) \cdot H'(Q, v, \gamma - \Delta)};$ 
22:   Compute for  $1 \leq j \leq \log N$ :

$$x^{(j)} = H(L_1^{(j)}, R_1^{(j)}), \quad y^{(j)} = H(L_2^{(j)}, R_2^{(j)}),$$

23:   Denote  $\mathbf{x} = [x^{(1)}, \dots, x^{(\log N)}], \quad \mathbf{y} = [y^{(1)}, \dots, y^{(\log N)}];$ 
24:   Compute for  $1 \leq i \leq N$ :

$$\zeta(i, j) = \begin{cases} 1 & \text{if the } j\text{-th bit of } (i-1) = 1 \\ -1 & \text{otherwise} \end{cases},$$


$$\phi_i = \prod_{1 \leq j \leq \log N} (x^{(j)})^{\zeta(i, j)}, \quad \psi_i = \prod_{1 \leq j \leq \log N} (y^{(j)})^{\zeta(i, j)},$$

25:   Denote  $\phi = [\phi_1, \dots, \phi_N], \quad \psi = [\psi_1, \dots, \psi_N];$ 
26:   if  $\mathbf{L}_1^{\mathbf{x}^2} P' \mathbf{R}_1^{\mathbf{x}^{-2}} = \mathbf{g}^{\alpha \cdot \phi} u^{\alpha \beta \cdot H'(P, u, \gamma)} \wedge \mathbf{L}_2^{\mathbf{y}^2} Q' \mathbf{R}_2^{\mathbf{y}^{-2}} = \mathbf{g}^{\delta \cdot \psi} u^{\delta \eta \cdot H'(Q, v, \gamma - \Delta)}$  then
27:     return 1 (accept).
28:   else
29:     return 0 (reject).
30:   end if
31: end procedure

```

In **Algorithm 1**. The system parameters \mathbf{pp} include a group \mathbb{G} of prime order p , with $\mathbf{g} = [g_1, \dots, g_n]$ and g_D as its generators, and two hash functions, H and H' , mapping from $\{0, 1\}^*$ to \mathbb{Z}_p . The argument system comprises a NIMA.PROOF algorithm and a NIMA.VERIFY algorithm. The NIMA.PROOF algorithm takes as input $(\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta, \alpha, \delta)$ and produces a proof $\Pi = (\mathbf{L}, \mathbf{R}, \alpha, \beta, \delta, \eta)$. The NIMA.VERIFY algorithm inputs $(\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta, \Pi)$ and outputs either 1 (accept) or 0 (reject).

The NIMA.PROOF involves a recursion procedure Pf , where $\beta = \prod_{j=1}^k (x_j + x_j^{-1}) 1^{\frac{N}{2^k}}$ and $\eta = \prod_{j=1}^k (y_j + y_j^{-1}) 1^{\frac{N}{2^k}}$ in the k -th recursion for x_j, y_j are the j -th outputs of H . The β and η are both known to the verifier and hence it is not require generators \mathbf{h} and \mathbf{s} to commit β and η in L_1, R_1, L_2, R_2 . Therefore, when proving the relation $\mathcal{R}_{\text{Minus}}$, we can set $\mathbf{h} = \mathbf{s} = 1^N$ and save nearly half of the exponentiations required during the recursion process, compared with directly extend the argument system for \mathcal{R}_{IP} in Bulletproofs to higher dimensions to prove the relation $\mathcal{R}_{\text{Dual}}$. Besides, it is sufficient for $\mathbf{r} = [g_D, \dots, g_D]$ with only one $g_D \in \mathbb{G}$ stored to compute the whole recursion, also reduces overhead.

Inherited from IP Argument [7] and Sum Argument [28], Minus Argument also satisfies the statistical witness-extended emulation [6,7] for non-trivial DL relation \mathbf{g}, u or a valid witness α , and g_D, v or a valid witness δ . Informally, it refers that given an adversary capable of generating an acceptable argument with a probability, there exists an emulator that can produce a argument with the same probability along with a witness w . Besides, the emulator is allowed to rewind the interaction between the prover and verifier to any previous move.

TripleRing-EC. We made Minus Argument (**Algorithm 1**) compatible with TripleRing-DL, yielding TripleRing-EC. To make it easier to read, we use the same color to represent the corresponding equations with (4). We observe that the underlying relation for verification of TripleRing generic construction is

$$\mathcal{R}_{\text{Verify-TR}} = \left\{ \begin{array}{l} ((Y, W, V_1, V_2, S_1, S_2) \in \mathbb{G}; (c_j, w_j) \in \mathbb{Z}_p) : \\ \quad Y \odot (V_1(z))^{-1} = \bigodot_{j=1}^N V_2(\mathbf{pk}_j, c_j \otimes w_j) \\ \wedge \quad W \odot (S_1(x))^{-1} = \bigodot_{j=1}^N S_2(\mathbf{pk}_D, w_j) = S_2(\mathbf{pk}_D, \bigotimes_{j=1}^N w_j) \\ \wedge \quad H(\mathbf{R}, \mathbf{pk}_D, M, Y, W) = \bigotimes_{j=1}^N c_j \end{array} \right\},$$

while the underlying relation for the verification of TripleRing-EC is

$$\mathcal{R}_{\text{Verify-EC}} = \left\{ \begin{array}{l} ((Y, W, V_1, V_2, S_1, S_2) \in \mathbb{G}; (c_j, w_j) \in \mathbb{Z}_p) : \\ \quad Y \cdot (V_1(z))^{-1} = \prod_{j=1}^N \mathbf{pk}_j^{c_j + w_j} \\ \wedge \quad W \cdot (S_1(x))^{-1} = \mathbf{pk}_D^{\sum_{j=1}^N w_j} \\ \wedge \quad H(\mathbf{R}, \mathbf{pk}_D, M, Y, W) = \sum_{j=1}^N c_j \end{array} \right\}.$$

Therefore, the Triple-EC runs following steps to sign/simulate a message M :

1. It runs the Sign/Sim algorithm in TripleRing-DL on input $(\mathbf{pp}, \mathbf{sk}_\pi, M, \mathbf{R}, \mathbf{pk}_D)$ / $(\mathbf{pp}, \mathbf{sk}_D, M, \mathbf{R}, \mathbf{pk}_D)$, and outputs a signature $\Sigma = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$. The intermediate values of the stage include $(Y_\pi, W_\pi, z_\pi, x_\pi)$.

2. It defines $\mathbf{g} = [\mathbf{pk}_1, \dots, \mathbf{pk}_n]$, $g_D = \mathbf{pk}_D$, $P = Y_\pi \odot (V_1(z_\pi))^{-1}$, $Q = W_\pi \odot (S_1(x_\pi))^{-1}$, $\alpha = [c_1 + w_1, \dots, c_N + w_N]$, $\delta = [w_1, \dots, w_N]$, $\Delta = \sum_{j=1}^N w_j$, $\gamma = \sum_{j=1}^N (c_j + w_j)$. Then, it choose generators $u, v \leftarrow_{\S} \mathbb{G}$ and runs NIMA.PROOF on input $(\{\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta\}, \alpha, \delta)$, and returns a proof Π .

Finally, the signature is $\Sigma = (r, Y_\pi, W_\pi, \Delta, \Pi)$. The designated verifier can verify follow these steps in order: (1) It computes $V_1(z_\pi)$ and $S_1(x_\pi)$ according to the recovery function $F(\mathbf{sk}_D, r)$; (2) It computes $c = H(\mathbf{R}, \mathbf{pk}_D, M, Y_\pi, W_\pi)$, $\gamma = c + \Delta$, $P = Y_\pi \odot (V_1(z_\pi))^{-1}$, $Q = W_\pi \odot (S_1(x_\pi))^{-1}$; (3) It runs NIMA.VERIFY on input $(\mathbf{pp}, \mathbf{g}, g_D, P, Q, \gamma, \Delta, \Pi)$ to verify the proof.

Correctness of TripleRing-EC follows that of TripleRing generic construction and NIMA. Next, we discuss its security.

Theorem 6. *TripleRing-EC satisfies unforgeability w.r.t. insider corruption if TripleRing-DL is unforgeability w.r.t. insider corruption and NIMA is statistical witness-extended emulation.*

Proof. We defer it to Appendix F. \square

Theorem 7. *TripleRing-EC satisfies signer anonymity if TripleRing-DL satisfies signer anonymity.*

Proof. We defer it to Appendix G. \square

Theorem 8. *TripleRing-EC satisfies non-transferability if TripleRing-DL satisfies non-transferability.*

Proof. We defer it to Appendix H. \square

According to Theorems 6, 7, 8 and results have been proven in Section 4.1, we have TripleRing-EC is a strong DVRS scheme that satisfies *unforgeability*, *signer anonymity*, and *non-transferability*.

Comparison with existing schemes. The signature of TripleRing-EC consists of $\Sigma = (r, Y_\pi, W_\pi, \Delta, \Pi)$. Here, $(r, Y_\pi, W_\pi, \Delta) \in \mathbb{G}^4 \times \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p$ and $\Pi = (\mathbf{L}, \mathbf{R}, \alpha, \beta, \delta, \eta) \in \mathbb{G}^{2 \log N} \times \mathbb{G}^{2 \log N} \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$, totaling $(4 \log N + 6)$ elements in \mathbb{G} and 5 elements in \mathbb{Z}_p . For $\lambda = 128$, the comparison with DL-based DVRS is shown in Table 1. As far as we know, TripleRing-EC is the first logarithmic-size DVRS scheme, which reduced size significantly.

Table 1. Comparison of Signature Sizes for DL-based DVRS schemes

| Scheme | # Elements in Signature | | Signature Sizes for Ring Sizes N | | | | Asymptotic Signature Sizes | Designated Verifier Property |
|------------------------------|----------------------------|------------------------------|------------------------------------|---------|----------|----------|----------------------------|------------------------------|
| | \mathbb{G} (33 Bytes) | \mathbb{Z}_p (32 Bytes) | 2^4 | 2^8 | 2^{12} | 2^{16} | | |
| [3] | 1 | $2N + 4$ | 1.1 KB | 16.2 KB | 256.2 KB | 4.0 GB | $O(N)$ | Weak |
| [4] | 1 | $3N + 1$ | 1.6 KB | 24.1 KB | 384.1 KB | 6.0 GB | $O(N)$ | Weak |
| TripleRing-EC (This work) | $4 \log N + 6$ | 5 | 0.9 KB | 1.4 KB | 1.9 KB | 2.4 KB | $O(\log N)$ | Strong |

4.2 TripleRing-LB: A Post-Quantum Instance from Lattice

Notations and Assumptions. For an odd modulus q , we denote by \mathbb{Z}_q the set $\{0, \dots, q-1\}$. The matrix \mathbf{A}^T is the transpose of \mathbf{A} , and \mathbf{I}_n is n -order identity matrix. Besides, $s \leftarrow_{\$} S$ means that s is chosen uniformly at random from the set S , and $x \leftarrow \chi$ means that x is chosen according to the distribution χ .

Definition 7 (Discrete Gaussian Distribution, [18]). *The Discrete Gaussian Distribution over \mathbb{Z}^m centers at $\mathbf{v} \in \mathbb{Z}^m$ with standard deviation σ is defined as $D_{\mathbf{v},\sigma}^m(\mathbf{x}) = \frac{\rho_{\mathbf{v},\sigma}^m(\mathbf{x})}{\rho_{\sigma}^m(\mathbb{Z}^m)}$, where $\rho_{\mathbf{v},\sigma}^m(\mathbf{x}) = (\frac{1}{\sqrt{2\pi\sigma^2}})^m e^{-\frac{\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}}$ is the Gaussian Function, $\mathbf{v} = \mathbf{0}$ could be omitted, and $\rho_{\sigma}^m(\mathbb{Z}^m) = \sum_{\mathbf{x} \in \mathbb{Z}^m} \rho_{\sigma}^m(\mathbf{x})$.*

Definition 8 (SIS_{q,n,m,d} Normal Form, [18]). *Given a random $\mathbf{A}' \leftarrow_{\$} \mathbb{Z}_q^{n \times (m-n)}$ and $\mathbf{A} = [\mathbf{I}_n \| \mathbf{A}']$, find a vector $\mathbf{s} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{s} = \mathbf{0}$ and $0 < \|\mathbf{s}\|_{\infty} \leq d$.*

Definition 9 (LWE_{q,n,m,\chi} Normal Form, [20]). *Given a random $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}$, and a probability distribution χ over \mathbb{Z}_q , distinguish $(\mathbf{A}^T, \mathbf{A}^T \mathbf{s} + \mathbf{e})$ and $(\mathbf{A}^T, \mathbf{v})$, where $\mathbf{s} \leftarrow \chi^N$, $\mathbf{e} \leftarrow \chi^m$ and $\mathbf{v} \leftarrow_{\$} \mathbb{Z}_q^m$.*

Lemma 1 ([19], Theorem 2). *Given integers $\bar{n} \geq 1, q \geq 2$ and $\bar{m} = O(\bar{n} \log q)$, there is a PPT algorithm $\text{TrapGen}(q, \bar{n}, \bar{m})$ that outputs a matrix $\mathbf{B} \leftarrow_{\$} \mathbb{Z}_q^{\bar{n} \times \bar{m}}$ and a trapdoor \mathbf{R}_B , such that the distribution of \mathbf{B} is $\text{negl}(\bar{n})$ -far from uniform. Moreover, for any $\mathbf{b} = \mathbf{B}^T \mathbf{s} + \mathbf{e}$, where $\mathbf{s} \in \mathbb{Z}_q^{\bar{n}}$ is arbitrary, and $0 < \|\mathbf{e}\| < q/O(\sqrt{\bar{n} \log q})$ or $\mathbf{e} \leftarrow D_{\sigma=\alpha q}^{\bar{m}}$ for small enough α , there is a polynomial time deterministic algorithm $\text{Invert}(\mathbf{R}_B, \mathbf{B}, \mathbf{b})$ that outputs \mathbf{s} and \mathbf{e} .*

Construction of TripleRing-LB. We now present the settings for a lattice-based Extended Type-T* Canonical Identification, which can be converted into the strong DVRS using TripleRing. We choose a classic SIS-based Canonical Identification [18], and an LWE function for the hidden function. The underlying assumptions include Definition 8 and 9, with rejection sampling techniques [18] to ensure that outputs reveal no information about the secret key.

- The public parameters include q, n, m, k, σ , and matrices $\mathbf{A}' \leftarrow_{\$} \mathbb{Z}_q^{n \times (m-n)}$, $\mathbf{A} = [\mathbf{I}_n \| \mathbf{A}']$. The space $\mathcal{S}_y = \mathbb{Z}_q^m$ has a communicative operation $\oplus = +$ in it, and the challenge space $\mathcal{S}_c = \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$ has communicative operations $\otimes = +$ and $\oslash = -$ in it.
 - For user i , the secret key is $\text{sk}_i = \mathbf{S}_i \leftarrow_{\$} \{-d, \dots, 0, \dots, d\}^{m \times k}$, and public key is $\text{pk}_i = \mathbf{T}_i$, such that $\mathbf{T}_i = \mathbf{A} \cdot \mathbf{S}_i$;
 - For the designated verifier, the secret key is $\text{sk}_D = (\mathbf{S}_D, \mathbf{R}_B)$, and public key is $\text{pk}_D = (\mathbf{T}_D, \mathbf{B})$, such that $\mathbf{S}_D \leftarrow_{\$} \{-d, \dots, 0, \dots, d\}^{m \times k}$, $\mathbf{T}_D = \mathbf{A} \cdot \mathbf{S}_D$. Besides, $(\mathbf{B}, \mathbf{R}_B)$ is a uniform matrix \mathbf{B} with trapdoor \mathbf{R}_B , generated according to Lemma 1 and used for hidden/recovery.
- Remarks.* The user i can also generate $(\mathbf{B}_i, \mathbf{R}_{B_i})$ and include it in the key-pairs to maintain consistency with the designated verifier. However, it is not required when generating a signature, so we omit it to simplify.

- The commit function $A(y) := \mathbf{A}\mathbf{y}$ for $y = \mathbf{y} \leftarrow_{\$} D_{\sigma}^m$, in $\text{Proof}_1(\text{sk})$;
- The hash function $H : \{0, 1\}^* \rightarrow \mathbf{c} \in \mathcal{S}_c = \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$;
- The response function $Z(\text{sk}, y, c) := \mathbf{S} \cdot \mathbf{c} - \mathbf{y}$, and we have $\mathbf{z} = \mathbf{S} \cdot \mathbf{c} - \mathbf{y}$;
- The verification function $V(\text{pk}, z, c) = V_1(z) + V_2(\text{pk}, c) = -\mathbf{A} \cdot \mathbf{z} + \mathbf{T} \cdot \mathbf{c}$;
- The simulation function $S(\text{pk}_D, x, w) = S_1(x) + S_2(\text{pk}_D, w) = -\mathbf{A} \cdot \mathbf{x} + \mathbf{T}_D \cdot \mathbf{w}$;
- Rejection Samplings: In Proof_2 , rejects and restarts at the first step of Proof_1 if $\|\mathbf{z}\|_{\infty} > 2\sigma\sqrt{m}$, where $\mathbf{z} = Z(\text{sk}, y, c)$. In Verify , rejects if $\|\mathbf{z}\|_{\infty} > 2\sigma\sqrt{m}$.
- The hidden function E and recovery function F . We set $\bar{n} = 2m$ and $\bar{m} = O(\bar{n} \log q)$. According to Lemma 1, the signer uses $\mathbf{B} \in \text{pk}_D$ and $(\mathbf{z}_{\pi}, \mathbf{x}_{\pi}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^m$ to compute $r = \mathbf{b} = \mathbf{B}^T \begin{bmatrix} \mathbf{z}_{\pi} \\ \mathbf{x}_{\pi} \end{bmatrix} + \mathbf{e}$. While the designated verifier can use its trapdoor $\mathbf{R}_B \in \text{sk}_D$ to recover $(\mathbf{z}_{\pi}, \mathbf{x}_{\pi})$ and proceed with the Verify .

We observe that the settings satisfy the following properties in Section 3.1.

1. The challenge space $\mathcal{S}_c = \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$ is a commutative group, with operation $\otimes = +$, while $\oslash = -$ is its inverse, under mod 3;
2. If \mathbf{c}_1 and \mathbf{c}_2 are uniformly distributed in \mathcal{S}_c , then $\mathbf{c}_1 \otimes \mathbf{c}_2$ is also uniformly distributed in \mathcal{S}_c . This also holds true for $\mathbf{w}_1 \otimes \mathbf{w}_2$ and $\mathbf{c}_1 \otimes \mathbf{w}_1$;
3. The verification function $V(\text{pk}, z, c) = -\mathbf{A} \cdot \mathbf{z} + \mathbf{T} \cdot \mathbf{c} = V_1(z) + V_2(\text{pk}, c)$ and simulation function $S(\text{pk}_D, x, w) = -\mathbf{A} \cdot \mathbf{x} + \mathbf{T}_D \cdot \mathbf{w} = S_1(x) + S_2(\text{pk}_D, w)$, where $\odot = +$ is a commutative group operation for their ranges;
4. The verification function $V_1(z) = -\mathbf{A} \cdot \mathbf{z}$ and simulation function $S_1(x) = -\mathbf{A} \cdot \mathbf{x}$ have additive homomorphism, i.e., $V_1(z_1) \odot V_1(z_2) = -\mathbf{A} \cdot \mathbf{z}_1 - \mathbf{A} \cdot \mathbf{z}_2 = -\mathbf{A} \cdot (\mathbf{z}_1 + \mathbf{z}_2) = V_1(z_1 \oplus z_2)$ and $S_1(x_1) \odot S_1(x_2) = -\mathbf{A} \cdot \mathbf{x}_1 - \mathbf{A} \cdot \mathbf{x}_2 = -\mathbf{A} \cdot (\mathbf{x}_1 + \mathbf{x}_2) = S_1(x_1 \oplus x_2)$, where $\oplus = +$ is a commutative group operation. Besides, the homomorphic operation can be efficiently computed;
5. Given sk matched with pk and c , there exists $\mathcal{I}_V(\text{sk}, c) = -\mathbf{S} \cdot \mathbf{c}$ such that $V_1(\mathcal{I}_V(\text{sk}, c)) = -\mathbf{A} \cdot (-\mathbf{S}) \cdot \mathbf{c} = \mathbf{A} \cdot \mathbf{S} \cdot \mathbf{c} = \mathbf{T} \cdot \mathbf{c} = V_2(\text{pk}, c)$. Meanwhile, given sk_D matched with pk_D and x , there exists $\mathcal{I}_S(\text{sk}_D, w) = -\mathbf{S}_D \cdot \mathbf{w}$ such that $S_1(\mathcal{I}_S(\text{sk}_D, w)) = -\mathbf{A} \cdot (-\mathbf{S}_D) \cdot \mathbf{w} = \mathbf{A} \cdot \mathbf{S}_D \cdot \mathbf{w} = \mathbf{T}_D \cdot \mathbf{w} = S_2(\text{pk}_D, w)$.

As a result, the above settings constitute a lattice-based instantiation of the Extended Type-T* Canonical Identification, satisfying the following security.

Theorem 9. *Under above settings, the lattice-based Extended Type-T* Canonical Identification is secure against special impersonation under key only attack in Definition 5, if $\text{LWE}_{q,n,m-n,\chi}$ and $\text{SIS}_{q,n,m+k,d}$ are hard, where $\chi = D_{\sigma}$ satisfies $\sigma = O(n^{1.5})$ and $d = 2\sigma(\kappa\sqrt{\log m} + 2\sqrt{m}) = \tilde{O}(n^{2.5})$.*

Proof. We defer it to Appendix I. □

The constrains for parameters are same with those in the pioneering work on lattice-based signatures [18]. According to TripleRing generic construction, the above Extended Canonical Identification can be converted into the strong DVRS scheme with *unforgeability*, *signer anonymity*, and *non-transferability*.

To the best of our knowledge, it represents the first strong DVRS scheme based on assumptions conjectured quantum-resistant (including previous schemes

that only met weak designated verifier property). Due to page limitations, we only provide settings based on the most common lattice assumptions and tools here, and leave discussions on sizes and further optimizations for future work.

5 Conclusion

In this paper, we discuss the primitive of Designated-Verifier Ring Signatures. After clarifying strong definitions, we present a generic construction, and two efficient instances. There are several appealing problems left for investigating:

1. **Enrich Instantiations.** In lattice-based settings, we only instantiated our generic construction using the most commonly used building blocks over standard lattice. In fact, employing existing techniques such as [2,13,26] can reduce both key sizes and signature sizes. Moreover, instantiation based on other post-quantum assumptions, such as code, would also be valuable.
2. **Extend to Multi Designated Verifier.** We focus on the single-verifier setting. However, in real-world applications, multi designated verifiers are always involved, as seen in the literature on signatures [9,29] and ring signatures [12]. Extending ours construction to multi-verifier would also be an interesting direction for future work.

Acknowledgments

The authors are grateful to the committees and reviewers of ICISC 2024. The authors are also grateful to Khoa Nguyen for his valuable discussions, and to the author of [9] for answering doubts. Jiaming Wen is partially supported by the CSC Visiting PhD Scholarship (No. 202306270167). Willy Susilo is partially supported by the Australian Laureate Fellowship (No. FL230100033) and the ARC Discovery Project (No. DP200100144). Yanhua Zhang is partially supported by the Key Foundation of Science and Technology Development of Henan Province (No. 242102211078), the Key Scientific Research Project of Higher Education of Henan Province (No. 24A520054), and the CSC Visiting Fellow Scholarship (No. 202308410421). Fuchun Guo is partially supported by the ARC Future Fellowship (No. FT220100046).

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: ASIACRYPT 2002. pp. 415–432. Springer (2002)
2. Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: CT-RSA 2014. pp. 28–47 (2014)
3. Balla, D., Behrouz, P., Grontas, P., Pagourtzis, A., Spyrahou, M., Vrettos, G.: Designated-verifier linkable ring signatures with unconditional anonymity. In: International Conference on Algebraic Informatics. pp. 55–68. Springer (2022)

4. Behrouz, P., Grontas, P., Konstantakatos, V., Pagourtzis, A., Spyrahou, M.: Designated-verifier linkable ring signatures. In: ICISC 2021. pp. 51–70. Springer (2021)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology* **22**(1), 114–138 (2009)
6. 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. pp. 327–357. Springer (2016)
7. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wulle, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: S&P 2018. pp. 315–334. IEEE (2018)
8. Chen, G., Wu, C., Han, W., Chen, X., Lee, H., Kim, K.: A new receipt-free voting scheme based on linkable ring signature for designated verifiers. In: International Conference on Embedded Software and Systems Symposia. pp. 18–23. IEEE (2008)
9. Damgård, I., Haagh, H., Mercer, R., Nitulescu, A., Orlandi, C., Yakubov, S.: Stronger security and constructions of multi-designated verifier signatures. In: TCC 2020. pp. 229–260. Springer (2020)
10. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31**(4), 469–472 (1985)
11. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: EUROCRYPT 1996. pp. 143–154. Springer (1996)
12. Kolby, S., Pagnin, E., Yakubov, S.: Multi designated verifier ring signatures. *IACR Communications in Cryptology* **1**(3) (2024)
13. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* **75**(3), 565–599 (2015)
14. Lee, J.S., Chang, J.H.: Strong designated verifier ring signature scheme. In: Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. pp. 543–547. Springer (2007)
15. Li, J., Wang, Y.: Universal designated verifier ring signature (proof) without random oracles. In: International Conference on Embedded and Ubiquitous Computing. pp. 332–341. Springer (2006)
16. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. *Journal of Cryptology* **36**(3), 23 (2023)
17. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In: PKC 2013. pp. 107–124. Springer (2013)
18. Lyubashevsky, V.: Lattice signatures without trapdoors. In: EUROCRYPT 2012. pp. 738–755. Springer (2012)
19. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT 2012. pp. 700–718. Springer (2012)
20. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005. pp. 84–93 (2005)
21. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT 2001. pp. 552–565. Springer (2001)
22. Saeednia, S., Kremer, S., Markowitch, O.: An efficient strong designated verifier signature scheme. In: ICISC 2003. pp. 40–54. Springer (2003)
23. Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology* **4**, 161–174 (1991)
24. Susilo, W., Zhang, F., Mu, Y.: Identity-based strong designated verifier signature schemes. In: ACISP 2004. pp. 313–324. Springer (2004)

25. Wen, J., Bai, L., Yang, Z., Zhang, H., Wang, H., He, D.: Larrs: Lattice-based revocable ring signature and its application for vanets. *IEEE Transactions on Vehicular Technology* **73**(1), 739–753 (2024)
26. Wen, J., Susilo, W., Yang, R., Yu, Z., Zhang, H.: Revocable ring signatures with cca-anonymity from standard lattices. *Computer Standards & Interfaces* **91**, 103893 (2025)
27. Yang, R., Au, M.H., Zhang, Z., Xu, Q., Yu, Z., Whyte, W.: Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In: *CRYPTO 2019*. pp. 147–175. Springer (2019)
28. 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*. pp. 251–281. Springer (2021)
29. Zhang, Y., Susilo, W., Chen, Y., Guo, F., Wen, J.: Lattice-based universal designated multi-verifiers signature scheme. In: *ISPEC 2024*. pp. 33–54. Springer (2024)

A Proof for Theorem 1

Proof. The proof of this theorem follows the Equation (1), (2), (3) in Section 3.2. We present more details there, where *Sim* is marked in cyan for simplicity.

The core of *Sign* (*Sim*) involves using V_2 and S_2 to aggregate public keys $\{\text{pk}_j\}_{j \neq \pi}$ and corresponding challenges and randomnesses into Y_{π}' and W_{π}' as the Response-Ring in **Fig. 2**. After obtained the challenge c' , it computes the missing $c_{\pi}' = c' \odot (\bigotimes_{j \neq \pi} c_j')$, where $c' = \bigotimes_{j=1}^N c_j'$ should hold. Then, the signer (*designated verifier*) computes z_{π} and r (w'_{π}, x'_{π} and r') by employing the Extended Type-T* Canonical Identification and c_{π}' .

During the verification, Y_{π}' and W_{π}' are reconstructed from all public keys and corresponding challenges and randomnesses. Then, the designated verifier computes the challenge c' , which should be equal to $\bigotimes_{j=1}^N c_j'$. Above all, correctness of TripleRing relies on that of Extended Type-T* Canonical Identification and hidden/recovery functions. It holds when these blocks are sound. \square

B Proof for Theorem 2

Proof. Suppose \mathcal{A} is a PPT algorithm that can break the unforgeability w.r.t. insider corruption of TripleRing, we use it to construct a PPT algorithm \mathcal{B} that can break the special impersonation under key only attack of its underlying Extended Type-T* Canonical Identification. Meanwhile, \mathcal{B} obtained the public parameters pp and a public key pk^* from its challenger \mathcal{C} .

Setup. \mathcal{B} chooses an index $i^* \leftarrow_{\$} [1, q_k]$, and generates $(\widetilde{\text{pk}}_i, \widetilde{\text{sk}}_i) \leftarrow \text{KeyGen}(\text{pp})$ for $i \in [1, q_k] \wedge i \neq i^*$ and $(\text{pk}_D, \text{sk}_D) \leftarrow \text{KeyGen}(\text{pp})$. \mathcal{B} sets $\widetilde{\text{pk}}_{i^*} = \text{pk}^*$, and returns pp and a set of public keys $S := \{\text{pk}_i\}_{i=1}^{q_k}$ and pk_D to \mathcal{A} .

Oracle Simulation. \mathcal{B} returns to the oracle queries as follows.

- **Random Oracle** $\mathcal{H}()$: On query, \mathcal{B} chooses $c \leftarrow_{\$} \mathcal{S}_c$ and returns c .
- **Corruption Oracle** $\mathcal{O}_{\text{Cor}}(\widetilde{\text{pk}}_i)$: On input a public key $\widetilde{\text{pk}}_i$, \mathcal{B} aborts if $i = i^*$. Otherwise, adds it to a corrupted set S_{Cor} and returns $\widetilde{\text{sk}}_i$.
- **Signing Oracle** $\mathcal{O}_{\text{Sign}}(R, \text{pk}_D, \widetilde{\text{pk}}_{\pi}, M)$: On input the public key ring $R = \{\text{pk}_1, \dots, \text{pk}_N\}$, the designated verifier's public key pk_D , the signer's public key $\widetilde{\text{pk}}_{\pi}$, and a message M , \mathcal{B} aborts if $\widetilde{\text{pk}}_{\pi} \notin R$. Otherwise, \mathcal{B} invokes the Corruption Oracle to obtain $\widetilde{\text{sk}}_{\pi}$, and returns Σ as follows:
 1. If $\pi \neq i^*$: $\Sigma \leftarrow \text{Sign}(R, \text{pk}_D, \widetilde{\text{sk}}_{\pi}, M)$;
 2. If $\pi = i^*$: \mathcal{B} chooses $z, \{c_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, \{w_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, x \leftarrow_{\$} \mathcal{S}_y$ to compute $Y = V_1(z) \odot \bigodot_{j=1}^N V_2(\text{pk}_j, c_j \otimes w_j), W = S_1(x) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j)$ and r . \mathcal{B} sets $H(R, \text{pk}_D, M, Y, W) = \bigotimes_{j=1}^N c_j$ in the random oracle, fails and aborts if the value has been set. Otherwise, $\Sigma = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$.
- **Simulation Oracle** $\mathcal{O}_{\text{Sim}}(R, \text{pk}_D, M)$: On input a message M , the public key ring $R = \{\text{pk}_1, \dots, \text{pk}_N\}$ and designated verifier's public key pk_D . \mathcal{B} invokes the Corruption Oracle to obtain sk_D and returns Σ as follows:
 1. If $\pi \neq i^*$: $\Sigma \leftarrow \text{Sim}(R, \text{pk}_D, \text{sk}_D, M)$;

2. If $\pi = i^*$: \mathcal{B} chooses $z', \{c'_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, \{w'_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, x' \leftarrow_{\$} \mathcal{S}_y$ to compute $Y = V_1(z') \odot \bigodot_{j=1}^N V_2(\text{pk}_j, c'_j \otimes w'_j), W = S_1(x') \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w'_j)$ and r' . \mathcal{B} sets $H(R, \text{pk}_D, M, Y, W) = \bigotimes_{j=1}^N c'_j$ in the random oracle, fails and aborts if the value has been set. Otherwise, $\Sigma = (r', \{c'_j\}_{j=1}^N, \{w'_j\}_{j=1}^N)$.

Challenge. When \mathcal{A} returns the corrupted set S_{Cor} and a forgery message-signature pair $(M^*, \Sigma^* = (r^*, \{c_j^*\}_{j=1}^N, \{w_j^*\}_{j=1}^N))$, \mathcal{B} fails and aborts if $\text{pk}^* \notin \text{S}_{\text{Cor}}$. Otherwise, $\text{pk}^* \in \text{S}_{\text{Cor}}$, \mathcal{B} uses the public key ring $R^* = \{\widetilde{\text{pk}}_{i_j}\}_{j=1}^N \subseteq \text{S}_{\text{Cor}}$ containing $\text{pk}^* = \widetilde{\text{pk}}_{i_{j^*}}$ to compute Y^*, W^* in **Verify**. \mathcal{B} rewinds to the point that M^*, R^*, Y^*, W^* is queried to H and returns a different c' instead. Then, \mathcal{B} will obtain another signature $\Sigma^\# = (r^\#, \{c_j^\#\}_{j=1}^N, \{w_j^\#\}_{j=1}^N)$, which satisfy:

$$\begin{aligned} Y^* &= V_1(z^*) \odot \bigodot_{j=1}^N V_2(\widetilde{\text{pk}}_{i_j}, c_j^* \otimes w_j^*) = V_1(z^\#) \odot \bigodot_{j=1}^N V_2(\widetilde{\text{pk}}_{i_j}, c_j^\# \otimes w_j^\#), \\ W^* &= S_1(x^*) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j^*) = S_1(x^\#) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j^\#). \end{aligned}$$

Since $\bigotimes_{j=1}^N c_j^* \neq \bigotimes_{j=1}^N c_j^\#$, there exists at least $c_j^* \neq c_j^\#$. If it is the index j^* that holds $c_{j^*}^* = c_{j^*}^\#$, aborts. With probability $\geq \frac{1}{N}$ we have $c_{j^*}^* \neq c_{j^*}^\#$, and

$$\begin{aligned} &V_1(z^*) \odot \bigodot_{j=1}^N V_2(\widetilde{\text{pk}}_{i_j}, c_j^* \otimes w_j^*) \\ &= V_1(z^* \oplus \mathcal{Z}_1^* \oplus \cdots \oplus \mathcal{Z}_{j^*-1}^* \oplus \mathcal{Z}_{j^*+1}^* \oplus \cdots \oplus \mathcal{Z}_n^*) \odot V_2(\text{pk}^*, c_{j^*}^* \otimes w_{j^*}^*) \\ &= V_1(\mathcal{Z}^*) \odot V_2(\text{pk}^*, c_{j^*}^* \otimes w_{j^*}^*) \\ &S_1(x^*) \odot \bigodot_{j=1}^N V_2(\widetilde{\text{pk}}_{i_j}, w_j^*) \\ &= V_1(x^* \oplus \mathcal{X}_1^* \oplus \cdots \oplus \mathcal{X}_{j^*-1}^* \oplus \mathcal{X}_{j^*+1}^* \oplus \cdots \oplus \mathcal{X}_n^*) \odot V_2(\text{pk}^*, w_{j^*}^*) \\ &= V_1(\mathcal{X}^*) \odot V_2(\text{pk}^*, w_{j^*}^*), \end{aligned}$$

where $\mathcal{Z}_i^* = \mathcal{I}_V(\widetilde{\text{sk}}_i, c_i^*), \mathcal{X}_i^* = \mathcal{I}_S(\widetilde{\text{sk}}_i, w_i^*)$ for $i \in [1, N] \wedge i \neq j^*$ and $\mathcal{Z}^* = z^* \oplus \mathcal{Z}_1^* \oplus \cdots \oplus \mathcal{Z}_{j^*-1}^* \oplus \mathcal{Z}_{j^*+1}^* \oplus \cdots \oplus \mathcal{Z}_n^*, \mathcal{X}^* = x^* \oplus \mathcal{X}_1^* \oplus \cdots \oplus \mathcal{X}_{j^*-1}^* \oplus \mathcal{X}_{j^*+1}^* \oplus \cdots \oplus \mathcal{X}_n^*$. Similarly, we can obtain $(c_{j^*}^\#, \mathcal{Z}^\#, \mathcal{X}^\#)$ from $\Sigma^\#$. Then, \mathcal{B} returns $(c_{j^*}^*, \mathcal{Z}^*, \mathcal{X}^*, c_{j^*}^\#, \mathcal{Z}^\#, \mathcal{X}^\#)$ to its challenger \mathcal{C} .

Probability Analysis. We analyze the probability of success in the above simulation, where $q_c, q_{\text{sig}}, q_{\text{sim}}, q_h$ are the number queries to the $\mathcal{O}_{\text{Cor}}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}, \mathcal{H}$.

\mathcal{O}_{Cor} : The probability of success in the first query is $\left(1 - \frac{1}{q_k}\right)$, and that of the second query is $\left(1 - \frac{1}{q_k}\right) \left(1 - \frac{1}{q_k-1}\right)$. After q_c queries, the probability of success is $\left(1 - \frac{1}{q_k}\right) \left(1 - \frac{1}{q_k-1}\right) \cdots \left(1 - \frac{1}{q_k-q_c+1}\right) = \frac{q_k-q_c}{q_k} = 1 - \frac{q_c}{q_k}$.

$\mathcal{O}_{\text{Sign}}$: The probability of success in the first query is at least $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right)$, and that of the second query is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right)$. After q_{sig} queries, the probability of success is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right) \cdots \left(1 - \frac{q_h+q_{\text{sig}}-1}{|\mathcal{S}_c|}\right) \geq 1 - \frac{q_{\text{sig}}(q_h+q_{\text{sig}}-1)}{|\mathcal{S}_c|}$.

\mathcal{O}_{Sim} : The probability of success in the first query is at least $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right)$, and that of the second query is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right)$. After q_{sim} queries, the probability of success is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right) \cdots \left(1 - \frac{q_h+q_{\text{sim}}-1}{|\mathcal{S}_c|}\right) \geq 1 - \frac{q_{\text{sim}}(q_h+q_{\text{sim}}-1)}{|\mathcal{S}_c|}$.

The probability of $\text{pk}^* \neq \widetilde{\text{pk}}_{i_{j^*}}$ in the challenge space \mathcal{S}_c is

$$\left(1 - \frac{1}{q_k - q_c}\right) \left(1 - \frac{1}{q_k - q_c - 1}\right) \cdots \left(1 - \frac{1}{q_k - q_c - N + 1}\right) = \frac{q_k - q_c - N}{q_k - q_c}.$$

Above all, if the probability of a successful forgery by \mathcal{A} is ϵ , then the probability of \mathcal{B} does not abort before rewinding is

$$\epsilon_{\mathcal{B}} := \epsilon \left(1 - \frac{q_c}{q_k}\right) \left(1 - \frac{q_{\text{sig}}(q_h + q_{\text{sig}} - 1)}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_{\text{sim}}(q_h + q_{\text{sim}} - 1)}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_k - q_c - N}{q_k - q_c}\right).$$

According to the Generalized Forking Lemma [28], the probability of a successful rewinding is at least $\frac{\epsilon_{\mathcal{B}}}{8}$ if $|\mathcal{S}_c| > 8q_h/\epsilon_{\mathcal{B}}$, and the reduction algorithm runs in time $T_R \cdot (8q_n/\epsilon_{\mathcal{B}}) \cdot \ln(8N/\epsilon_{\mathcal{B}})$ if \mathcal{A} runs in time T_R . As a result, we have $c_{j^*}^* \neq c_{j^*}^\#$ with probability at least $1/N$, and the probability for \mathcal{B} to break the special impersonation is $\epsilon' \geq \frac{\epsilon}{8} \left(1 - \frac{q_c}{q_k}\right) \left(1 - \frac{q_{\text{sig}}(q_h + q_{\text{sig}} - 1)}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_{\text{sim}}(q_h + q_{\text{sim}} - 1)}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_k - q_c - N}{q_k - q_c}\right)$. Here, $|\mathcal{S}_c| > \max\{q_{\text{sig}}(q_h + q_{\text{sig}} - 1), q_{\text{sim}}(q_h + q_{\text{sim}} - 1)\}$ is required, and a successful forgery is constructed. \square

C Proof for Theorem 3

Proof. We present how to construct an algorithm \mathcal{B} , which achieves perfect signer anonymity against full key exposure in the random oracle model.

Setup. \mathcal{B} runs $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ to obtain public parameters pp . Then, \mathcal{B} generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KeyGen}(\text{pp}; r_i)$ using randomness r_i for $i \in [1, q_k]$, and $(\text{pk}_D, \text{sk}_D) \leftarrow \text{KeyGen}(\text{pp})$. \mathcal{B} returns pp and a set of public keys $\mathcal{S} := \{\text{pk}_i\}_{i=1}^{q_k}$ and pk_D to the algorithm \mathcal{A}_1 .

Oracle Simulation. \mathcal{B} returns to the oracle queries as follows.

- **Random Oracle $\mathcal{H}()$:** On query, \mathcal{B} chooses $c \leftarrow_{\mathcal{S}} \mathcal{S}_c$ and returns c .
- **Signing Oracle $\mathcal{O}_{\text{Sign}}(\mathbf{R}, \text{pk}_D, \text{pk}_\pi, M)$:** On input the public key ring $\mathbf{R} = \{\text{pk}_1, \dots, \text{pk}_N\}$ and designated verifier's public key pk_D , signer's public key pk_π , and a message M , \mathcal{B} returns $\Sigma \leftarrow \text{Sign}(\mathbf{R}, \text{pk}_D, \text{sk}_\pi, M)$.
- **Simulation Oracle $\mathcal{O}_{\text{Sim}}(\mathbf{R}, \text{pk}_D, M)$:** On input the public key ring $\mathbf{R} = \{\text{pk}_1, \dots, \text{pk}_N\}$ and designated verifier's public key pk_D and a message M , \mathcal{B} returns $\Sigma' \leftarrow \text{Sim}(\mathbf{R}, \text{pk}_D, \text{sk}_D, M)$.

Challenge and Output. \mathcal{A}_1 gives \mathcal{B} a message M^* , a public key ring $R^* \subseteq S$ with $\text{pk}_{i_0}, \text{pk}_{i_1} \in R^*$. \mathcal{B} chooses $z, \{c_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, \{w_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, x \leftarrow_{\$} \mathcal{S}_y$ to compute $Y = V_1(z) \odot \bigodot_{j=1}^N V_2(\text{pk}_j, c_j \otimes w_j), W = S_1(x) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j)$. \mathcal{B} sets $H(R, \text{pk}_D, M, Y, W) = \bigotimes_{j=1}^N c_j$ in the random oracle, aborts if the value has been set. Otherwise, $\Sigma = (z, x, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$. \mathcal{B} returns Σ along with randomnesses $\{r_j\}_{j=1}^{q_k}$ to \mathcal{A}_2 , and picks a random bit b . Finally, \mathcal{A}_2 outputs a bit b' . Since b is not involved in the generation of the signature Σ , \mathcal{A}_2 can only succeed with a probability of one-half.

Probability Analysis. We analyze the probability of success in the above simulation, where $q_{\text{sig}}, q_{\text{sim}}, q_h$ are the number queries to the $\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}, \mathcal{H}$.

$\mathcal{O}_{\text{Sign}}$: The probability of success in the first query is at least $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right)$, and that of the second query is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right)$. After q_{sig} queries, the probability of success is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right) \cdots \left(1 - \frac{q_h+q_{\text{sig}}-1}{|\mathcal{S}_c|}\right) \geq 1 - \frac{q_{\text{sig}}(q_h+q_{\text{sig}}-1)}{|\mathcal{S}_c|}$.

\mathcal{O}_{Sim} : The probability of success in the first query is at least $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right)$, and that of the second query is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right)$. After q_{sim} queries, the probability of success is $\left(1 - \frac{q_h}{|\mathcal{S}_c|}\right) \left(1 - \frac{q_h+1}{|\mathcal{S}_c|}\right) \cdots \left(1 - \frac{q_h+q_{\text{sim}}-1}{|\mathcal{S}_c|}\right) \geq 1 - \frac{q_{\text{sim}}(q_h+q_{\text{sim}}-1)}{|\mathcal{S}_c|}$.

Here, $|\mathcal{S}_c| > \max\{q_{\text{sig}}(q_h + q_{\text{sig}} - 1), q_{\text{sim}}(q_h + q_{\text{sim}} - 1)\}$ is required, and \mathcal{B} does not abort, indicating that no unbounded adversary can achieve a non-negligible advantage greater than random guessing. \square

D Proof for Theorem 4

Proof. We prove that the distributions of $\text{Sign}(\text{pk}_\pi, M, R, \text{pk}_D)$ and $\text{Sim}(M, R, \text{pk}_D)$ are identical for a given message M , public key ring R , and designated verifier's public key pk_D . This ensures perfect non-transferability.

Setup. \mathcal{B} runs $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ to obtain public parameters pp . Then, \mathcal{B} generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KeyGen}(\text{pp})$ for $i \in [1, q_k]$ and $(\text{pk}_D, \text{sk}_D) \leftarrow \text{KeyGen}(\text{pp})$. \mathcal{B} returns pp and a set of public keys $S := \{\text{pk}_i\}_{i=1}^{q_k}$ and pk_D to the algorithm \mathcal{A} .

Oracle Simulation. \mathcal{B} responses queries for $\mathcal{H}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}$ as the Theorem 3.

Challenge and Output. \mathcal{A} gives \mathcal{B} a message M^* , a public key ring $R^* \subseteq S$, and designated verifier's public key pk_D , and \mathcal{B} returns $\Sigma_b \in \{\Sigma_0, \Sigma_1\}$. Consider the distributions of $\Sigma_0 = \text{Sign}(R, \text{pk}_D, \text{sk}_\pi, M) = (r, \{c_j\}_{j=1}^N, \{w_j\}_{j=1}^N)$ and $\Sigma_1 = \text{Sim}(R, \text{pk}_D, \text{sk}_D, M) = (r', \{c'_j\}_{j=1}^N, \{w'_j\}_{j=1}^N)$, where $r = E(\text{pk}_D, z_\pi, x_\pi)$ and $r' = E(\text{pk}_D, z'_\pi, x'_\pi)$.

Since $y_\pi, \{c_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c, \{w_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_c, \{x_j\}_{j=1}^N \leftarrow_{\$} \mathcal{S}_y$ during the generation of Σ_0 , and $\eta, z'_\pi, \{c'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c, \{w'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_c$, and $\phi, \{x'_j\}_{j \neq \pi} \leftarrow_{\$} \mathcal{S}_y$

during the generation of Σ_1 , we have the following indistinguishable relations.

$$\begin{aligned} \{c_j\}_{j \neq \pi} &\approx \{c'_j\}_{j \neq \pi}, \quad c \approx c', \quad c_\pi = c \odot \left(\bigotimes_{j \neq \pi} c_j \right) \approx c' \odot \left(\bigotimes_{j \neq \pi} c'_j \right) = c'_\pi; \\ \{w_j\}_{j \neq \pi} &\approx \{w'_j\}_{j \neq \pi}, \quad w_\pi = \eta \odot c'_\pi \approx w'_\pi, \text{ since } \eta, c'_\pi, w'_\pi \text{ are all uniform;} \\ z_\pi &\approx z'_\pi, \text{ since } z_\pi = Z(\text{sk}_\pi, y_\pi, c_\pi \otimes w_\pi) \text{ with uniform } y_\pi, c_\pi \otimes w_\pi, \text{ and } z'_\pi \leftarrow_{\$} \mathcal{S}_c; \\ x_\pi &\approx x'_\pi, \text{ since } x'_\pi = Z(\text{sk}_D, \phi, w'_\pi) \text{ with uniform } \phi, w'_\pi, \text{ and } x_\pi \leftarrow_{\$} \mathcal{S}_y; \\ r &= E(\text{pk}_D, z_\pi, x_\pi) \approx r' = E(\text{pk}_D, z'_\pi, x'_\pi) \end{aligned}$$

As a result, the unbounded adversary \mathcal{A} cannot distinguish Σ_0 and Σ_1 , and can only succeed with the probability of one-half.

Probability Analysis. The probability of success in the above simulation is the same as Theorem 3. It requires $|\mathcal{S}_c| > \max\{q_{sig}(q_h + q_{sig} - 1), q_{sim}(q_h + q_{sim} - 1)\}$, where q_{sig}, q_{sim}, q_h are the number queries to the $\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}, \mathcal{H}$. Then, \mathcal{B} does not abort, indicating that no unbounded adversary can achieve a non-negligible advantage greater than random guessing. \square

E Proof for Theorem 5

Proof. Suppose that the adversary \mathcal{A} can break 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} returns $\text{pp} = (\mathbb{G}, p, g)$ and $\text{pk} = y$ to \mathcal{A} .

\mathcal{A} returns $(c^{(1)}, z^{(1)}, c^{(2)}, z^{(2)})$, where $c^{(1)} \neq c^{(2)}$, and satisfies

$$g^{z^{(1)}} \cdot \text{pk}^{c^{(1)}} = g^{z^{(2)}} \cdot \text{pk}^{c^{(2)}}, \quad g^{x^{(1)}} \cdot \text{pk}_D^{w^{(1)}} = g^{x^{(2)}} \cdot \text{pk}_D^{w^{(2)}}.$$

As a result, \mathcal{B} can extract the secret key $\text{sk} = \frac{z^{(1)} - z^{(2)}}{c^{(2)} - c^{(1)}}$ and $\text{sk}_D = \frac{x^{(1)} - x^{(2)}}{w^{(2)} - w^{(1)}}$ as the solution to the DL problem, which is contradicted with Definition 6. \square

F Proof for Theorem 6

Proof. Suppose that \mathcal{A} is a PPT adversary breaking the unforgeability w.r.t. insider corruption of TripleRing-EC, we use it to construct a PPT algorithm \mathcal{B} that can break the unforgeability of TripleRing-DL.

Setup. \mathcal{B} is given the system parameter pp' and a set of public keys \mathcal{S} from the challenger of TripleRing-DL, and picks a random generator $u \leftarrow_{\$} \mathbb{G}$ and returns $\text{pp} = (\text{pp}', u)$ to the adversary \mathcal{A} .

Oracle Simulation. \mathcal{B} responses queries for *Random Oracle* $\mathcal{H}()$ with $c \leftarrow_{\$} \mathcal{S}_c$. \mathcal{B} responses queries for *Signing Oracle* $\mathcal{O}_{\text{Sign}}$ and *Simulation Oracle* \mathcal{O}_{Sim} as follows. It firstly uses the corresponding oracles in TripleRing-DL to obtain $\Sigma^\# = (r^\#, \{c_j^\#\}_{j=1}^N, \{w_j^\#\}_{j=1}^N)$, and runs the *Verify* algorithm of TripleRing-DL to obtain $Y_\pi^\# = V_1(z_\pi^\#) \odot \bigodot_{j=1}^N V_2(\text{pk}_j, c_j^\# \otimes w_j^\#)$, $W_\pi^\# = S_1(x_\pi^\#) \odot \bigodot_{j=1}^N S_2(\text{pk}_D, w_j^\#)$, $c = c_1^\# + \dots + c_N^\#$, $\Delta = w_1^\# + \dots + w_N^\#$. Then, \mathcal{B} runs the NIMA.PROOF to obtain the proof $\pi^\#$. Finally, \mathcal{B} returns $\Sigma^\# = (r^\#, Y_\pi^\#, W_\pi^\#, \Delta^\#, \pi^\#)$.

Challenge. In the challenge phase, the adversary \mathcal{A} returns a signature $\Sigma^* = (r^*, Y_\pi^*, W_\pi^*, \Delta^*, \Pi^*)$ w.r.t. a message M^* and the public key ring R^* . According to the statistical witness-extended emulation of NIMA, \mathcal{B} could run an extractor \mathcal{E} to extract $(c_1^* + w_1^*, \dots, c_N^* + w_N^*)$, and (w_1^*, \dots, w_N^*) such that $P^* = Y_\pi^* \odot (V_1(z_\pi^*))^{-1} = \bigodot_{j=1}^N V_2(\text{pk}_j, c_j^* + w_j^*)$, and $Q^* = W_\pi^* \odot (S_1(x_\pi^*))^{-1} = \bigodot_{j=1}^N S_2(\text{pk}_D, w_j^*)$. Then, \mathcal{B} computes (c_1^*, \dots, c_N^*) and returns the signature $\Sigma^* = (r^*, \{c_j^*\}_{j=1}^N, \{w_j^*\}_{j=1}^N)$ to the challenger of TripleRing-DL.

Above all, the unforgeability w.r.t. insider corruption of TripleRing-EC can be reduced to that of TripleRing-DL. \square

G Proof for Theorem 7

Proof. Suppose that \mathcal{A} is a PPT adversary breaking the signer anonymity of TripleRing-EC, we use it to construct a PPT algorithm \mathcal{B} that can break the signer anonymity of TripleRing-DL.

Setup. \mathcal{B} is given the system parameter pp' and a set of public keys S from the challenger of TripleRing-DL, and picks a random generator $u \leftarrow_{\mathcal{S}} \mathbb{G}$ and returns $\text{pp} = (\text{pp}', u)$ to the adversary \mathcal{A} .

Oracle Simulation. \mathcal{B} responses queries for $\mathcal{H}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}$ as in Theorem 6.

Challenge and Output. In the challenge phase, \mathcal{A}_1 gives $(M^*, R^*, \text{pk}_{i_0}, \text{pk}_{i_1})$ to \mathcal{B} . Similar with in Theorem 3, \mathcal{B} sends them to its challenger and uses the received $(r^*, \{c_j^*\}_{j=1}^N, \{w_j^*\}_{j=1}^N)$ to compute $\Sigma^* = (r^*, Y^*, W^*, \Delta^*, \Pi^*)$ via the signing algorithm of TripleRing-EC, and sends Σ^* along with randomnesses $\{r_j^*\}_{j=1}^{q_k}$ to \mathcal{A}_2 . Finally, \mathcal{A}_2 outputs a bit b' , which could be used by \mathcal{B} to send to its challenger and break the anonymity of TripleRing-DL.

Above all, the signer anonymity of TripleRing-EC can be reduced to that of TripleRing-DL. \square

H Proof for Theorem 8

Proof. Suppose that \mathcal{A} is a PPT adversary breaking the signer anonymity of TripleRing-EC, we use it to construct a PPT algorithm \mathcal{B} that can break the signer anonymity of TripleRing-DL.

Setup. \mathcal{B} is given the system parameter pp' and a set of public keys S from the challenger of TripleRing-DL, and picks a random generator $u \leftarrow_{\mathcal{S}} \mathbb{G}$ and returns $\text{pp} = (\text{pp}', u)$ to the adversary \mathcal{A} .

Oracle Simulation. \mathcal{B} responses queries for $\mathcal{H}, \mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Sim}}$ as in Theorem 6.

Challenge and Output. In the challenge phase, \mathcal{A} gives $(M^*, R^*, \text{pk}_{i_0}, \text{pk}_{i_1})$ to \mathcal{B} . Similar with in Theorem 4, \mathcal{B} sends them to its challenger and uses the received $(r^*, \{c_j^*\}_{j=1}^N, \{w_j^*\}_{j=1}^N)$ and $(r'^*, \{c_j'^*\}_{j=1}^N, \{w_j'^*\}_{j=1}^N)$ to compute $\Sigma_0^* = (r^*, Y^*, W^*, \Delta^*, \Pi^*)$ via the signing algorithm of TripleRing-EC and $\Sigma_1^* = (r'^*, Y'^*, W'^*, \Delta'^*, \Pi'^*)$ via the simulation algorithm of TripleRing-EC. Then, \mathcal{B} 's challenger returns $\Sigma_b^* \in \{\Sigma_0^*, \Sigma_1^*\}$ to \mathcal{B} , and \mathcal{B} forward it to \mathcal{A} . Same as in Theorem 4, the distributions of Σ_0^* and Σ_1^* are indistinguishable.

Above all, the non-transferability of TripleRing-EC can be reduced to that of TripleRing-DL. \square

I Proof for Theorem 9

Proof. Suppose that the adversary \mathcal{A} can break the special impersonation under key only attack. We use it to construct a polynomial time algorithm \mathcal{B} that can break either $\text{LWE}_{q,n,m-n,\chi}$ Normal Form or $\text{SIS}_{q,n,m+k,d}$ Normal Form, where $\chi = D_\sigma$ satisfies $\sigma = O(n^{1.5})$, and $d = 2\sigma(\kappa\sqrt{\log m} + 2\sqrt{m}) = \tilde{O}(n^{2.5})$.

\mathcal{B} is given the parameters q, n, m, k, σ , and matrices $\mathbf{A}' \leftarrow_{\$} \mathbb{Z}_q^{n \times (m-n)}$ and $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{n \times k}$. Then, it has $\mathbf{A} = [\mathbf{I}_n \| \mathbf{A}'] \in \mathbb{Z}_q^{n \times m}$ and $\hat{\mathbf{A}} = [\mathbf{A} \| \mathbf{U}] \in \mathbb{Z}_q^{n \times (m+k)}$.

\mathcal{B} chooses $\mathbf{E} \leftarrow D_\sigma^{n \times k}$, therefore, each column of $\mathbf{A}^T \cdot \mathbf{E} = [\mathbf{I}_n \| \mathbf{A}']^T \cdot \mathbf{E}$ is a $\text{LWE}_{q,n,m-n,\chi}$ instance for $\chi = D_\sigma$. According to Definition 9, it is computationally indistinguishable from a random element from $\mathbb{Z}_q^{m \times k}$. Then, \mathcal{B} sets

$\text{pk} = \mathbf{T} = \hat{\mathbf{A}} \cdot \mathbf{E}' = \mathbf{A} \cdot \mathbf{E} + \mathbf{U}$, where $\mathbf{E}' = \begin{bmatrix} \mathbf{E} \\ \mathbf{I}_k \end{bmatrix}$ such that $\|\mathbf{E}'\|_\infty \leq \sigma\sqrt{\log m}$.

Finally, \mathcal{B} returns $\text{pp} = (q, n, m, d, \sigma, \mathbf{A}, \text{pk})$ to \mathcal{A} .

\mathcal{A} returns $(\mathbf{c}^{(1)}, \mathbf{z}^{(1)}, \mathbf{c}^{(2)}, \mathbf{z}^{(2)})$, where $\mathbf{c}^{(1)} \neq \mathbf{c}^{(2)}$, and satisfies

$$-\mathbf{A} \cdot \mathbf{z}^{(1)} + \mathbf{T} \cdot \mathbf{c}^{(1)} = -\mathbf{A} \cdot \mathbf{z}^{(2)} + \mathbf{T} \cdot \mathbf{c}^{(2)}$$

Combining it with $\mathbf{T} = \hat{\mathbf{A}} \cdot \begin{bmatrix} \mathbf{E} \\ \mathbf{I}_k \end{bmatrix}$ we have

$$\hat{\mathbf{A}} \cdot \begin{bmatrix} \mathbf{E} \\ \mathbf{I}_k \end{bmatrix} \cdot (\mathbf{c}^{(1)} - \mathbf{c}^{(2)}) = \mathbf{T} \cdot (\mathbf{c}^{(1)} - \mathbf{c}^{(2)}) = \mathbf{A} \cdot (\mathbf{z}^{(1)} - \mathbf{z}^{(2)}) = \hat{\mathbf{A}} \cdot \begin{bmatrix} \mathbf{z}^{(1)} - \mathbf{z}^{(2)} \\ \mathbf{0} \end{bmatrix}$$

Namely, $\mathbf{s} = \begin{bmatrix} \mathbf{E} \\ \mathbf{I}_k \end{bmatrix} \cdot (\mathbf{c}^{(1)} - \mathbf{c}^{(2)}) - \begin{bmatrix} \mathbf{z}^{(1)} - \mathbf{z}^{(2)} \\ \mathbf{0} \end{bmatrix}$ is a solution for $\hat{\mathbf{A}} \cdot \mathbf{s} = \mathbf{0}$, while

\mathbf{s} is also a non-zero vector (At least, the last k rows of \mathbf{s} is $(\mathbf{c}^{(1)} - \mathbf{c}^{(2)}) \neq \mathbf{0}$).

Besides, according to chosen criteria and rejection samplings, we have the matrix $\mathbf{E} \in D_\sigma^{n \times k}$ such that $\|\mathbf{E}\|_\infty \leq \sigma\sqrt{\log m}$ with high probability, challenges $\mathbf{c}^{(1)}, \mathbf{c}^{(2)} \in \mathcal{S}_c = \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$, and vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}$ satisfy $\|\mathbf{z}^{(1)}\|_\infty \leq 2\sigma\sqrt{m}, \|\mathbf{z}^{(2)}\|_\infty \leq 2\sigma\sqrt{m}$. As a result, there is a norm bound

$$\|\mathbf{s}\|_\infty \leq 2\kappa\sigma\sqrt{\log m} + 4\sigma\sqrt{m} = 2\sigma(\kappa\sqrt{\log m} + 2\sqrt{m})$$

Here, according to [18], we have $\sigma = 12 \cdot d \cdot \kappa \cdot \sqrt{m}$, and $\kappa < k < n, m = 2n$ for the Normal Form. As a result, \mathbf{s} is a solution for $\text{SIS}_{q,n,m+k,d}$ Normal Form, where $d = 2\sigma(\kappa\sqrt{\log m} + 2\sqrt{m}) = \tilde{O}(n^{2.5})$, which is contradicted with its hardness. \square