DualRing-PRF: Post-Quantum (Linkable) Ring Signatures from Legendre and Power Residue PRFs

Xinyu Zhang[1,2], Ron Steinfeld[1], Joseph K. Liu[1], Muhammed F. Esgin[1], Dongxi Liu[2], and Sushmita Ruj[3]

[1] Monash University, Melbourne, AU
{Xinyu.Zhang1,Ron.Steinfeld,Joseph.Liu,Muhammed.Esgin}@monash.edu
[2] Data61, CSIRO, Sydney, AU
Dongxi.Liu@csiro.au
[3] University of New South Wales
Sushmita.Ruj@unsw.edu

Abstract. Ring signatures are one of the crucial cryptographic primitives used in the design of privacy-preserving systems. Such a signature scheme allows a signer to anonymously sign a message on behalf of a spontaneously formed group. It not only ensures the authenticity of the message but also conceals the true signer within the group. An important extension of ring signatures is *linkable* ring signatures, which prevent a signer from signing twice without being detected (under some constraints). Linkable ring signatures offer advantages in applications where full anonymity might jeopardise the intended purpose, such as privacy-oriented cryptocurrencies like Monero.

In this work, we introduce post-quantum ring signature (DualRing-PRF) and linkable ring signature (DualRing$_L$-PRF) schemes whose security solely rely on symmetric-key primitives (namely, Legendre PRF and power residue PRF). Our construction of the ring signature departs from previous approaches with similar security assumptions, offering the most competitive signature sizes for small and medium-sized rings. In particular, for a ring size of 16, DualRing-PRF has a communication overhead 1.4 times smaller than the state-of-the-art scheme proposed by Goel et al. [PETS'21]. Furthermore, we demonstrate the extension of DualRing-PRF to incorporate linkability and non-slanderability. Compared to the existing *one-time* traceable ring signature (a variant of linkable ring signature) by Scafuro and Zhang [ESORICS'21], our construction supports many-time signing and achieves significantly smaller signature sizes when ring size exceeds 16. This advantage becomes more pronounced as the ring size increases.

Keywords: Ring Signature, Linkability, Post-Quantum, Symmetric Key Primitives

1 Introduction

A ring signature scheme allows a signer to sign a message while preserving anonymity behind a spontaneously chosen group, commonly known as a “ring”. A verifier can check the validity of the signature but cannot identify the actual signer among all possible ring members. The notion was initially formalized in [37], and ring signatures, along with its related notions such as ad-hoc identification have been extensively studied since then [1, 17, 4, 39, 11, 24, 13, 2, 23].

The ring signature scheme stands out as a pivotal cryptographic tool in the development of privacy-preserving systems. The initial motivation was to facilitate anonymous disclosure of secrets (also known as whistleblowing). For example, a government agency can employ the scheme to sign information with respect to the ring of all agents. The signed information can then be verified as originating from a reputable source without exposing the actual signer.

An important extension of a ring signature scheme is linkability, a feature that prevents a signer from discreetly signing twice without being detected. Linkable ring signatures [30] are especially beneficial in applications where full anonymity could compromise the intended purpose, such as in e-voting [41]. In fully anonymous voting systems, for instance, a voter can potentially accumulate the necessary minimum number of distinct votes by repeatedly submitting its votes. Linkable ring signatures provide an efficient method of detecting such fraudulent multi-voting behaviors.

Most existing ring signatures and linkable ring signatures depend on security assumptions like large integer factorisation and discrete logarithm, which are vulnerable to quantum attacks (e.g., Shor’s algorithm [40]). Given the wide range of privacy-oriented applications, it is imperative to develop post-quantum counterparts for (linkable) ring signature schemes that are both concretely efficient and secure. There has recently been a sequence of works that construct quantum-resistant ring signatures by utilizing lattice-based cryptography [31, 19, 43, 33, 32, 18, 7] and symmetric-key primitives such as block ciphers and hash functions [28, 16, 38, 22]. While lattice-based ring signatures have a smaller communication overhead, schemes based on symmetric-key primitives rely on arguably weaker security assumptions, such as collision-resistant hash functions, pseudorandom functions, and block ciphers, providing more confidence in their security.

1.1 Our Contributions

Contribution I. DualRing-PRF: Post-Quantum Ring Signature. We present a novel construction of a post-quantum ring signature based on symmetric-key primitives, characterized by the smallest communication overhead compared to state-of-the-art schemes with similar security assumptions for small and medium-sized rings (i.e., rings of size from 2 to 1000).[4] Our protocol extends the DualRing framework [43] to accommodate Legendre PRF and power

[4] Efficiency of ring signatures for small and medium-sized rings is crucial in real-world applications due to their inherent limitations. That is, for a ring with size

Ring Size	$n = 2^4$	2^6	2^{10}	2^{11}	**Assumption**	Link	NS
[7]	31	32	34	34.5	Lattice-based Assumptions	✓	✓
[18]	< 11	11	< 18	< 18		✓	✗
[33]	< 16	< 18	18	< 19		✗	✗
[43]	4	5	20	36		✗	✗
[32]	< 13	13	< 14	< 14		✗	✗
[16]	597	838	1352	1440	Low-MC	✗	✗
[28]	~ 175	$\sim$243	$\sim$388	$\sim$404		✗	✗
[22]	48	51	57	58		✗	✗
[38]*	32	131	2048	4096	Hash	✓*	✓*
DualRing-PRF	34	37	56	74	Legendre/Power Residue PRFs	✗	✗
DualRing$_L$-PRF	42	45	64	82		✓	✓

Table 1. Signature Size (in KB) Comparison Among Post-Quantum (Linkable/Traceable) Ring Signatures based on Symmetric-Key Primitives for 128-bit Security. [38] is one-time *traceable* ring signature.

residue PRF in a non-trivial manner. The extension stems from our observation of the modular design inherent in the key identification scheme based on Legendre/power residue PRFs. Specifically, it involves a three-move identification protocol with *non-negligible* soundness error, complemented by a zero-knowledge proof to enhance the overall soundness of the three-move identification process.

However, in contrast to Type-T* canonical identification schemes (e.g., Schnorr identification) which are straightforwardly supported by DualRing, the additional zero-knowledge proof potentially reveals the identity of the real signer. To address this challenge, we propose a novel solution using the cut-and-choose [28] technique and seamlessly integrate it with MPC-in-the-head [26] based zero-knowledge proofs. As shown in Table 1, our design yields the most competitive signature size among all symmetric-key primitives based ring signatures for ring sizes ranging from 16 to 1024.

Contribution II. Linkable DualRing-PRF. We extend DualRing-PRF to a linkable ring signature scheme (DualRing$_L$-PRF) that achieves linkability and non-slanderability. The properties are crucial for applications which require efficient detection of double signing, where a malicious user attempts to sign two messages with respect to the same event. Our approach leverages the versatility of MPC-in-the-head [26] and optimizes it for Legendre/power residue PRFs based signatures. Consequently, our scheme features a 4KB linking tag and adds 4 - 12KB to the signature size. Comparing our construction with the state-of-the-art one-time traceable ring signature [38] (see Table 1), our linkable ring

N, (linkable) ring signatures require at least N operations for both signing and verification, as well as the storage of N public keys. These limitations restrict the size of the ring, making it challenging to scale. On the other hand, for very small ring sizes, such as 2 to 10, the anonymity guarantee is too weak. For instance, after the release of Monero version 0.13, the smallest ring size was fixed at 16 to ensure transaction anonymity. Therefore, as stated in [43], one could argue that the most relevant ring size in practice falls between 10 and 2000.

signature exhibits significantly shorter signatures for ring sizes exceeding 16. This advantage becomes more pronounced as ring sizes increase.

Contribution III. Implementations. We implemented DualRing-PRF using Python with a single thread and benchmarked its performance on a MacBook Pro equipped with the Apple M1 MAX chip. Our implementation offers three different parameter sets for both power residue and Legendre PRF-based ring signatures, allowing for a trade-off between signature size and running time. Despite the inherent performance degradation of Python, we demonstrate that for generating a ring signature with a ring size of 256, the power-residue PRF-based DualRing-PRF requires 26 to 88 seconds while the Legendre PRF-based scheme requires 55 - 183 seconds.

1.2 Technical Overview

Building Blocks Our construction involves three building blocks, DualRing [43], MPC-in-the-head [26], and LegRoast [8].

DualRing DualRing is a generic ring signature construction with the formation of two rings: a ring of commitments and a ring of challenges. It can be seen as a compiler which compiles a special type of Σ-protocol, called the Type-T* canonical identification scheme, to a ring signature. In particular, the Type-T* canonical identification (e.g., Schnorr's identification) is a three-move public-key authentication protocol whose verification algorithm V can be divided to two algorithms V_1 and V_2 such that $V = V_1 \odot V_2$ for some group operation $\odot$, where V_1 is additive or multiplicative homomorphic. The benefit of using the DualRing compiler is that the compiled ring signature consists of a single response and n challenges for a ring of size n, unlike in [1], where the ring signature comprises a single challenge and n responses.

MPC-in-the-Head. MPC-in-the-head is a technique utilized for constructing public-coin zero knowledge proofs of knowledge using MPC protocols. In a nutshell, the prover simulates an N-party functionality representing an NP relation $R(x, w)$ in the head. Here, x denotes the statement hard-wired into the circuit, while w is the witness secretly shared among virtual parties. The function outputs 1 if and only if $R(x, w) = 1$. After the simulation, the prover sends the verifier commitments to each virtual parties' input shares, secret random tapes and communicated messages during the simulation of the MPC protocol (commonly referred to as views). Additionally, it sends the output shares of the parties, which should reconstruct to 1. Subsequently, the verifier randomly chooses a subset of commitments to be opened and verifies that the committed messages are consistent with an honest execution of the MPC protocol according to the opened input shares, random tapes, and previously received output shares.

LegRoast. LegRoast/PorcRoast is a post-quantum signature scheme which enables a signer to prove the knowledge of a secret key in relation to the output of Legendre/power residue PRF using the MPC-in-the-head paradigm. Let p be a large odd prime and t be an integer such that $t|p-1$. The t-th power residue PRF (defined in Section 2.1) is a one-way function denoted by $\mathcal{L}^t_K(\cdot) : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{Z}_t$

(note when $t = 2$, it is the Legendre PRF). Owing to the limitation that the PRF only produces $\log_2(t)$ bit of output, the public key should consist of many PRF evaluations denoted as $pk = (pk_1, \ldots, pk_L) = (\mathcal{L}_K(I_1), \ldots, \mathcal{L}_K(I_L))$ for some fixed arbitrary list $\mathcal{I} = (I_1, \ldots, I_L)$. The (relaxed) key identification problem can thus be described as follows: given a public key consists of L Legendre PRF outputs, prove the knowledge of a value K' such that for a large fraction of $B \leq L$ symbols, $\mathcal{L}_{K'}(I_b) = \mathcal{L}_K(I_b)$ for all $b \in [B]$ simultaneously.

In LegRoast, the key identification scheme is constructed by applying the MPC-in-the-head based zero-knowledge proof that demonstrates the existence of correctly computed o_b, where $o_b = (K' + I_b)r_b$ for some randomness $r_b \in \mathbb{F}_p$, such that $\mathcal{L}_0^t(o_b) = pk_{I_b} + \mathcal{L}_0^t(r_b)$ for all $b \in [B]$. Note that $\{I_b\}_{b\in[B]}$ are a random fraction of $\mathcal{I}$ that is randomly chosen by the verifier once the prover commits to K' and $\{r_b\}_{b\in[B]}$. The completeness holds owing to the multiplicative homomorphism of the PRFs: $\mathcal{L}_0^t(a \cdot b) = \mathcal{L}_0^t(a) + \mathcal{L}_0^t(b)$, and the soundness of the key identification protocol follows the soundness of the underlying MPC-in-the-head. The corresponding signature scheme is obtained by applying Fiat-Shamir transformation to the identification scheme.

Our Approach We now outline our approaches for constructing efficient (linkable) ring signatures based on the aforementioned building blocks. Our main observation is that the key identification scheme based on Legendre/power-residue Pseudo-Random Functions (PRFs) in LegRoast [8] can be modularized. Namely, the scheme comprises two segments:

1. A three-move protocol that is reminiscent to the Type-T* canonical identification scheme.
2. An MPC-in-the-head-based zero-knowledge proof demonstrating the correct computation of the response generated by the three-move protocol.

In essence, by employing the DualRing compiler, we can transform the Legendre, power-residue PRF-based Type-T* canonical identification scheme into a DualRing-type ring signature. Leveraging the fact that DualRing generates a single response, we only need to validate it using a single MPC-in-the-head-based zero-knowledge proof. Since the size of the underlying MPC-in-the-head contributes to the most of the signature size, this compilation process will lead to a slight increase in communication complexity compared to the original LegRoast signature.

However, the compiled ring signature loses its anonymity because the verifier of the resulting ring signature must use the real signer's challenge from the Type-T* canonical identification scheme to validate the MPC-in-the-head-based zero-knowledge proof. To tackle this issue, we propose a novel solution that utilizes the well-known cut-and-choose technique [28] to securely conceal the real signer's challenges in the zero-knowledge proof.

At a high-level, we have the real signer splits all challenges generated from the Type-T* canonical identification into secret shares. These shares are then utilized by the signer to construct the MPC-in-the-head proof. Consequently, rather than employing the challenge itself, the verifier can use the shares to authenticate the proof, thereby ensuring the anonymity of the signer.

However, the signer must additionally convince the verifier that the shares of challenges are generated honestly. This can be achieved by allowing the signer to generate more challenge shares than required. Subsequently, the signer accumulates and commits to the seeded permutation of shares using some vector commitment scheme (e.g., Merkle tree). Upon receiving the commitments of these shares, the verifier requests the signer to disclose a random subset of these shares along with its seed for permutation. Given that the challenges are public values, the verifier can readily verify whether the shares are computed honestly. The remaining unopened shares can then be used in the MPC-in-the-head protocol.

Post-Quantum Linkable Ring Signature. The underlying idea of linkable ring signature schemes is to allow the signer to compute a linking tag (denoted as `tag`) using both the event ID and the secret key. A crucial requirement is to ensure that the tag is computed using the same key as employed in the signature. Compiling any MPC-in-the-head-based ring signatures into linkable ring signatures is straightforward, as MPC-in-the-head serves as a generic method for proving arbitrary circuits. In other words, let $R(w, x_1, \ldots, x_\ell)$ be the NP relation that is proven by the MPC-in-the-head in ring signature schemes, and $R(w', \mathtt{tag})$ be the NP relation with secret input w' and public input `tag`. Then, an MPC-in-the-head based linkable ring signature scheme proves the following relation:

$$(R(w, x_1, \ldots, x_\ell) = 1) \wedge (R(w', \mathtt{tag}) = 1) \wedge (w = w') \tag{1}$$

Although the construction adds a constant overhead to the original ring signature scheme, its straightforward application can result in a circuit size that is at least twice as large, leading to significant communication overhead. We demonstrate that by reusing the witness shares and combining the responses with a random linear combination, the linkable ring signature can be constructed with only a small increase (4 - 12 KB) in the signature size.

1.3 Related Works

There are two commonly used approaches for building ring signature schemes.[5]

Accumulator-based Ring Signature. The approach, exemplified by the work in [17], typically combines cryptographic accumulators with one-way domains and zero-knowledge proofs. The main advantage of this approach is the ability to achieve sublinear signature sizes relative to the size of the ring. However, many existing accumulators require a trusted setup, which is often undesirable.

[5] We have selected these two paradigms because they are commonly employed in the development of symmetric-key primitives based post-quantum ring signatures, which is relevant to this work. For a more comprehensive survey of ring signatures, we refer the reader to [12] and [9].

On the other hand, accumulators with transparent setups, such as Merkle tree-based accumulators, can result in considerably large signatures, despite their good asymptotic performance.

Σ-Protocol based Ring Signature. The approach, represented by [1], allows the signer to generate $n-1$ "decoy" signatures sequentially and closes the ring by creating a "real" signature using the secret key. Generally, this paradigm does not necessitate a trusted setup and can be more efficient than accumulator-based ring signatures for small-size rings. However, the signature size is linearly dependent on the ring size, making the scheme impractical when the ring size is large. Recent advancements have addressed the limitation. Notably, "Stacking Sigmas" [23] achieved an logarithmic signature size in this paradigm by utilizing partially binding vector commitments. Another work, DualRing [43], enhanced the paradigm by leveraging the special properties of many well-known Σ-protocols (e.g., Schnorr's identification and GQ identification [25]), which are called *Type-T^* canonical identification schemes*. Consequently, the signature contains ℓ challenges and a single response, as opposed to ℓ responses and a single challenge in the original scheme [1], where ℓ denotes the size of the ring. Such improvement is profound since in many Σ-protocols (especially post-quantum ones), the response is much larger than the challenges.

Due to the absence of algebraic structures in symmetric-key primitives, [16, 28] followed the first design approach, which leverages accumulators with one-way domain to construct ring signature schemes. The high-level idea is to enable the signer to prove to the verifier its knowledge of a pre-image of an accumulated value in zero-knowledge. As such, their work heavily relies on accumulators (e.g., Merkle tree) and zero-knowledge proofs for generic circuits. Concretely, given a set of ℓ public keys $\mathcal{PK} = pk_1, \ldots, pk_\ell$, the signer computes the accumulated value $\texttt{acc}$ by arranging the public keys as the leaves of a Merkle tree. Notably, for every $pk_j \in \mathcal{PK}$, there exists a unique membership proof $\texttt{auth}_j$. The signer then utilizes an MPC-in-the-head based zero-knowledge proof to demonstrate that for some $\pi \in [\ell]$, it has the knowledge of the secret key sk_π corresponding to a public key pk_π, and that $\texttt{auth}_\pi$ is a valid membership proof with respect to $\texttt{acc}$. Anonymity is ensured due to the zero-knowledge property of the underlying proof system, while the unforgeability is guaranteed by the (knowledge) soundness. It is worth noting that the construction introduced in [16] assumes black-box access to the zero-knowledge proof, allowing one to improve the protocol by applying a more efficient MPC-in-the-head proof [28]. A similar construction can also be found in [10], which presents ID-based ring signature schemes.

While the previously mentioned schemes exhibit logarithmic communication complexity, their concrete performance falls below satisfactory standards. The situation arises since verifying a Merkle membership proof in the MPC-in-the-head framework entails evaluating hash functions in the circuit, which is expensive. Goel et al. [22] addressed the problem by moving Merkle membership proof verification *outside* of the MPC circuit using the cut-and-choose technique, resulting in a significant decrease in signature sizes.

Scafuro and Zhang[38] introduced a *one-time* (the security such as unforgeability, anonymity, and non-frameability is guaranteed only if the signer uses the secret key at most once) traceable ring signature scheme by leveraging Naor's bit commitment scheme and its trapdoor variant [36]. The underlying idea of this scheme is that knowing the trapdoor of the commitment scheme enables the signer to equivocate the commitment to any given string. Specifically, the signer, denoted S_π, selects a set of $\ell - 1$ random strings and assigns the value of x_π to some garbage value. Then, the signer generates $n-1$ binding commitments to $(x_j)_{j\in[n]/\{\pi\}}$ and an equivocal commitment to x_π. Subsequently, the signer proceeds to evaluate the random oracle on input of the message, the ring, and the commitments $(c_1, \ldots, c_\ell)$, yielding a challenge z. In order for the signature to be accepted, the signer must show that the bit-wise XOR of $\{x_1, \ldots, x_n\}$ is equal to z. In pursuit of this, the signer equivocates x_π to x'_π such that $z = x_1 \oplus \ldots \oplus x'_\pi \oplus \ldots \oplus x_n$ and shows to the verifier that x_j is a valid opening of c_j for all $j \in [\ell]$.

2 Preliminaries

Notations. We denote by λ the security parameter. We say that a function $\texttt{negl} : \mathbb{N} \to \mathbb{N}$ is negligible if, for every positive polynomial $p(\cdot)$ and all sufficiently large integer z, it holds that $\texttt{negl}(z) < 1/p(z)$. We denote by $[d]$ the set of integers $\{1, \ldots, d\}$. For a set A, let $a \overset{\$}{\leftarrow} A$ denote that a is uniformly chosen at random from A. The notation $[\cdot]$ stands for additive secret-shared values with full threshold, and $[\cdot]_i$ for the i-th share held by MPC-in-the-head party P_i. Our signature schemes make use of Merkle tree based accumulators, which are formally presented in Appendix A.

2.1 Legendre PRF and Power Residue PRF

The Legendre symbol is a multiplicative function that maps an element $a \in \mathbb{F}_p$ to the value of $0, 1$, or -1, depending on whether a is a quadratic residue modulo p.

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a = b^2 \pmod{p} \text{ for some } b \in \mathbb{F}_p^* \\ 0, & \text{if } a = 0 \pmod{p} \\ -1, & \text{otherwise} \end{cases}$$

The (keyed) Legendre PRF is then defined by mapping the Legendre symbol with a secret shift K to $\{0, 1\}$ as follows:

$$\mathcal{L}_K(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{K+a}{p}\right)\right)\right\rfloor \in \{0, 1\}.$$

It is easy to see that the Legendre PRF $\mathcal{L}_K$ defined above has multiplicative homomorphism. Namely, for non-zero $a, b \in \mathbb{F}_p^*$,

$$\mathcal{L}_0(a \cdot b) = \mathcal{L}_0(a) + \mathcal{L}_0(b).$$

To improve PRF throughput, Damgård proposed to use higher power residue symbols instead of quadratic residue symbols [15]. Specifically, for an integer t and a prime p such that $t|(p-1)$, compute the t-th power residue symbol $a \mapsto a^{(p-1)/t} \mod p$, yielding a $\log_2(t)$-bits output per PRF call, in contrast to the single-bit output obtained by the Legendre PRF. The definition of the t-th power residue PRF is shown as follows.

Definition 1 (t-th Power Residue PRF). *Let p be an odd prime and t be an integer such that $p \equiv 1 \mod t$. Let $g \in \mathbb{F}_p^*$ be the generator. We define the t-th power residue symbol $\mathcal{L}_K^t(\cdot) : \mathbb{F}_p \to \mathbb{Z}_t$ with the hidden shift K as*

$$\mathcal{L}_K^t(a) = \begin{cases} i, & \text{if } (a+K) \not\equiv 0 \mod p \text{ and } (a+K)/g^i = z^t \mod p \\ 0, & \text{if } (a+K) \equiv 0 \mod p \end{cases}$$

It is worth noting that when $t = 2$, the power residue PRF is equivalent to the Legendre PRF.

To simplify the notation, we extend the definitions of the Legendre PRF and power residue PRF to be 'list-wise'. Namely, given a list $\mathcal{I} = \{I_1, \ldots, I_L\} \in \mathbb{F}_p^L$, denote the PRF evaluations as $\mathcal{L}_K^t(\mathcal{I}) = (\mathcal{L}_K^t(I_1), \ldots, \mathcal{L}_K^t(I_L)) \in \mathbb{Z}_t^L$. For the sake of completeness, we incorporate the (relaxed) NP-relations for the Legendre PRF and power residue PRF as defined in [8] into our work.

Definition 2 (Legendre / t-th Power Residue PRF Relation [8]). *For an odd prime p, a positive integer t with $t|(p-1)$, and a list $\mathcal{I} = (I_1, \ldots, I_L)$, where $I_\ell \in \mathbb{F}_p$ for all $\ell \in [L]$, we define the Legendre / t-th power residue PRF relation $R_{\mathcal{L}^t}$ as*

$$R_{\mathcal{L}^t} = \{(\mathcal{L}_K^t(\mathcal{I}), K) \in \mathbb{Z}_t^L \times \mathbb{F}_p | K \in \mathbb{F}_p\}.$$

In order to facilitate the development of efficient signature schemes that prove knowledge of the secret key in the context of the Legendre PRF / k-th power residue PRF, Beullens et al. [8] introduced a relaxed version (Definition 3) of the original relation.

Definition 3 (β-approximate PRF relation ([8])). *Given $\beta \in [0,1]$, an odd prime p, a positive integer $t|(p-1)$, and a list $\mathcal{I}$ of L elements uniformly chosen from $\mathbb{F}_p$ at random, we define the β-approximate PRF relation $R_{\beta\mathcal{L}^t}$ as*

$$R_{\beta\mathcal{L}} = \{(s, K) \in \mathbb{Z}_t^L \times \mathbb{F}_p | \exists a \in \mathbb{Z}_t : d_H(s + (a, \ldots, a), \mathcal{L}_K^t(\mathcal{I})) \leq \beta L\}$$

where $d_H(\cdot, \cdot)$ denotes the Hamming distance.

According to [8, Theorem 1], the β-approximate Legendre PRF relation is as hard as Legendre PRF relation with the probability at least $1-2p\cdot\Pr[\mathfrak{B}(L,1/2+1/\sqrt{p}+2/p)\geq(1-\beta)L]$ over the choice of $\mathcal{I}$, where $\mathfrak{B}(n,q)$ denotes the binomial distribution with n samples each with success probability q. Simply put, with the proper choice of the parameter L (sufficiently large) and β (sufficiently small) [8], if there exists a PPT algorithm that is able to find the β-approximate witness K, then K is also a witness of the exact Legendre PRF relation.

Security of Legendre/Power Residue PRFs. It is known that if the adversary is allowed to query the PRF in the quantum mechanism superposition, the hidden shift can be determined in polynomial-time, using a single query [14, 42]. However, if the oracle can only be queried classically, there exists a memoryless collision search algorithm with $O(\sqrt{p}\log p)$ Legendre symbol evaluations when $\sqrt{p}\log p$ queries are available [29]. Recent works concurrently improved the attack, either by exploiting the multiplicative property of the Legendre function [6, 27] or by allowing input-independent pre-computations [34]. Frixons and Schrottenloher [21] studied the quantum algorithms for the hidden shift problem with respect to the Legendre PRF. In particular, their attack can reach a time complexity $2^{O(\sqrt{\log_2 p})}\sqrt[3]{p}$, while using $\sqrt[3]{p}$ classical queries and a sub-exponential number of qubits memory.

Above attacks cannot be applied to key recovery of signature schemes based on the Legendre PRF (and power residue PRF) such as LegRoast [8]. The main reason is that the adversary can only obtain the public key, which is a small number of uncontrolled random outputs of the PRF. Thus, it is infeasible for the adversary to query the PRF at the superposition, making the attacks presented in [14, 42] inapplicable. In addition, attacks showed in [21, 34] require a massive number of classical queries ($O(\sqrt[3]{p})$), which is significantly larger than the number of revealed public key symbols. Thus, we may only consider attacks that have a limited number of queries.

When L is less than the optimal number of queries, the memoryless collision search presented in [29] results in the attack with time complexity $O(\frac{p\log p}{L})$, while the table-based collision search showed in [6, 27] has run time $O(\frac{p\log^2 p}{L^2})$ and $O(\frac{p\log p\log\log p}{L^2})$, respectively. These complexities are exponential in relation to the security parameter, especially when L is significantly smaller than p. It is crucial to emphasise that the enhancements proposed in [6, 27] are reliant on the sequential characteristics of Legendre PRF queries. Thus, it remains an open problem whether these attacks can be extended to random point queries.

For power residue symbols, the possibly best known attack is due to [6], in which the number of queries is bounded by $L\leq\sqrt[4]{p}$, and the time complexity is $O(\frac{p\log^2 p}{L^2\log^2 t})$ power residue symbol evaluations with $O(L^2\log t)$ memory, especially for small t. A better attack when t is a large value with run time $O(\frac{p\log^2 p}{Lt\log^2 t})$ and $O(L\log t)$ memory.

2.2 Ring Signature and Linkable Ring Signature

Ring Signature. We begin the section by introducing the syntax of a ring signature scheme. Let $\mathbb{PK} = (pk_1, pk_2, pk_3, \dots)$ denote the set of all public keys added to the system and a ring of size ℓ is denoted as $\mathcal{PK} \subseteq \mathbb{PK}$ such that $|\mathcal{PK}| = \ell$.

Definition 4 (Syntax of Ring Signature). *A* ring signature *scheme is a tuple of PPT algorithms* $(\texttt{Setup}, \texttt{KGen}, \texttt{Sign}, \texttt{Verify})$ *with message space* $\mathcal{M}$ *that are described as follows:*

– $\texttt{pp} \leftarrow \texttt{Setup}(1^\lambda)$*: On input a security parameter* λ *in unary, outputs a set of public parameters denoted as* $\texttt{pp}$*.*
– $(sk, pk) \leftarrow \texttt{KGen}(\texttt{pp})$*: On input the public parameters generated from* $\texttt{Setup}$*, outputs a public key* pk *and the corresponding secret key* sk*. From now on, public parameters are implicit inputs to the following algorithms.*
– $\sigma \leftarrow \texttt{Sign}(\mathcal{PK}, m, sk_\pi)$*: On input a list* $\mathcal{PK} \subseteq \mathbb{PK}$ *of* ℓ *public keys, a message* $m \in \mathcal{M}$*, and a secret key* sk_π *corresponds to* pk_π *where* $pk_\pi \in \mathcal{PK}$*, outputs a signature* σ*.*
– $0/1 \leftarrow \texttt{Verify}(\mathcal{PK}, m, \sigma)$*: On input a list* $\mathcal{PK}$ *of* ℓ *public keys, a message* m *and a ring signature* σ*, outputs a bit* $0/1$ *indicating the validity of* σ *on the message* m *with respect to the ring* $\mathcal{PK}$*.*

Definition 5 (Perfect Completeness). $(\texttt{Setup}, \texttt{KGen}, \texttt{Sign}, \texttt{Verify})$ *are (perfectly) complete if for adversary* $\mathcal{A}$*, it holds:*

$$\Pr\left[1 = \texttt{Verify}(\mathcal{PK}, m, \sigma) \;\middle|\; \begin{array}{l} \texttt{pp} \leftarrow \texttt{Setup}(1^\lambda) \\ \{sk_i, pk_i\}_{i\in[\ell]} \leftarrow \texttt{KGen}(\texttt{pp}) \\ (\mathcal{PK}, m) \leftarrow \mathcal{A}(\texttt{pp}, \{sk_i, pk_i\}_{i\in[\ell]}) \\ \sigma \leftarrow \texttt{Sign}(\mathcal{PK}, m, sk_\pi) \textit{ with} \\ pk_\pi \in \mathcal{PK} \end{array} \right] = 1$$

Before defining unforgeability and anonymity, we introduce the following oracles, which together model the ability of the adversaries in breaking the security of the schemes.

- $pk_i \leftarrow \mathcal{JO}(\perp)$: The joining oracle, on request, adds a new user to the system. It returns the public key $pk_i \in \mathcal{PK}$ of the new user.
- $sk_i \leftarrow \mathcal{CO}(pk_i)$: The corruption oracle. On input a public key $pk_i \in \mathcal{PK}$ that is a query output of $\mathcal{JO}$, returns the corresponding private key sk_i.
- $\sigma \leftarrow \mathcal{SO}(\mathcal{PK}, m, \pi)$: On input a list of ℓ public keys $\mathcal{PK}$, a message $m \in \mathcal{M}$, and the signer index π, returns a valid signature σ.

Our security proofs are in the random oracle model. Hence, random oracles are also simulated. We now define the unforgeability game as follows. **Unforgeability.** Unforgeability is defined as the following game between the extractor $\mathcal{B}$ and the adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}, \mathcal{CO}, \mathcal{SO}$ and the random oracle.

1. $\mathcal{B}$ generates and gives $\mathcal{A}$ the system parameters $\texttt{pp}$.
2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.
3. $\mathcal{A}$ gives $\mathcal{B}$ a set $\mathcal{PK}$ of ℓ public keys, a message $m \in \mathcal{M}$, and a signature σ.

$\mathcal{A}$ wins the game if

1. $\texttt{Verify}(\mathcal{PK}, m, \sigma) = 1$;
2. all of the public keys in $\mathcal{PK}$ are query outputs of $\mathcal{JO}$;
3. no public keys in $\mathcal{PK}$ have been input to $\mathcal{CO}$; and
4. σ is not a query output of $\mathcal{SO}$.

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda) = \Pr[\mathcal{A} \text{ wins the unforgeability game}]$$

Definition 6 (Unforgeability w.r.t. Insider Corruption). *For any polynomial time adversary $\mathcal{A}$, a ring signature satisfies unforgeability w.r.t. inside corruption if $\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda)$ is negligible.*

Anonymity. Anonymity is defined as the following game between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$, in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$ and the random oracle along with all randomnesses to generate secret keys ($\mathcal{A}$ can generate all signatures with the secret keys, hence it does not need the oracle access to $\mathcal{SO}$).

1. $\mathcal{B}$ generates and gives ($\mathcal{A}$) the system parameters $\texttt{pp}$.
2. $\mathcal{A}$ may query $\mathcal{JO}$ and random oracle according to any adaptive strategy (note that $\mathcal{A}$ knows all secret keys).
3. $\mathcal{A}$ gives $\mathcal{B}$ a set $\mathcal{PK}$ of ℓ public keys (can contain adversarally generated public keys), a message $m \in \mathcal{M}$, two indices $i_0, i_1 \in [\ell]$. $\mathcal{B}$ randomly selects a bit $b \in \{0, 1\}$ and computes $\sigma \leftarrow \texttt{Sign}(\mathcal{PK}, m, sk_{i_b})$ and returns σ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{anon}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

Definition 7 (Anonymity against Full Key Exposure). *For any polynomial time adversary $\mathcal{A}$, a ring signature is anonymous against full key exposure if $\mathbf{Adv}_{\mathcal{A}}^{anon}(\lambda)$ is negligible.*

A linkable ring signature scheme consists of an additional $\texttt{Link}$ algorithm, which determines whether two signatures are generated by the same signer with respect to the same event. In addition to the verification completeness defined in the ring signature, a linkable ring signature scheme requires linking completeness: if two signatures are signed for the same event according to specification, then they are linked if and only if the two signatures share a common signer. On top of unforgeability and anonymity, a linkable ring signature should also satisfy linkability and non-slanderability. Namely, linkability requires that it is infeasible for a signer to generate two signatures correspond to the same event ID such that they are determined to be unlinked. Non-slanderability ensures that no signer can generate a signature which is determined to be linked with another signature that is not generated by the signer. In other words, it prevents adversaries from framing honest users.

Linkable Ring Signature. A linkable ring signature scheme adds a `Link` algorithm to the syntax definition of ring signature. Furthermore, the `Sign` and `Verify` algorithms takes an additional input of an event ID e_{ID}. We show the syntax definition of a linkable ring signature as follows.

Definition 8 (Syntax Definition of Linkable Ring Signature). *A linkable ring signature scheme, is a tuple of five algorithms, include* $(\texttt{Setup}, \texttt{KGen}, \texttt{Sign}, \texttt{Verify}, \texttt{Link})$.

- $\texttt{pp} \leftarrow \texttt{Setup}$*: Same as* `Setup` *in Definition 4.*
- $(sk, pk) \leftarrow \texttt{KGen}$*: Same as* `KGen` *in Definition 4.*
- $\sigma \leftarrow \texttt{Sign}(\mathcal{PK}, m, sk_\pi, e_{ID})$*: On input a set of ℓ public keys, a message $m \in \mathcal{M}$, a secret key sk_π, and an event ID e_{ID}, produces a signature σ.*
- $0/1 \leftarrow \texttt{Verify}(\mathcal{PK}, m, \sigma, e_{ID})$*: On input a set of ℓ public keys, a message $m \in \mathcal{M}$, a signature σ and an event ID e_{ID}, outputs 1 if the signature is valid. Otherwise, outputs* 0*.*
- $0/1 \leftarrow \texttt{Link}((\mathcal{PK}_1, m_1, \sigma_1), (\mathcal{PK}_2, m_2, \sigma_2), e_{ID})$*: On input two sets of ℓ_1 and ℓ_2 public keys $\mathcal{PK}_1, \mathcal{PK}_2$, two messages $m_1, m_2 \in \mathcal{M}$, two valid signatures (σ_1, σ_2), and an event ID e_{ID}, outputs* 1 *if unlinked. Otherwise,* 0*.*

Linkability Linkability is defined in the following game between the challenger $\mathcal{B}$ and an adversary $\mathcal{A}$ in which $\mathcal{A}$ is given $\mathcal{JO}, \mathcal{CO}, \mathcal{SO}$ and the random oracle:

1. $\mathcal{B}$ generates and gives $\mathcal{A}$ the system parameters `pp`.
2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.
3. $\mathcal{A}$ gives $\mathcal{B}$ an event ID e_{ID}, two sets of ℓ_1, ℓ_2 public keys $\mathcal{PK}_1$ and $\mathcal{PK}_2$, messages m_1, m_2 and signatures σ_1, σ_2.

$\mathcal{A}$ wins the game if

1. all public keys in $\mathcal{PK}_1 \cup \mathcal{PK}_2$ are query outputs of $\mathcal{JO}$.
2. $\texttt{Verify}(\mathcal{PK}_i, m_i, \sigma_i, e_{ID}) = 1$ for $i = 1, 2$ such that σ_i are not outputs of $\mathcal{SO}$;
3. $\mathcal{CO}$ has been queried less than two times (i.e., $\mathcal{A}$ can have at most one secret key); and
4. $\texttt{Link}(\sigma_1, \sigma_2) = 1$ (i.e., two signatures are unlinked).

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{link}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}]$$

Definition 9 (Linkability). *For any polynomial time adversary $\mathcal{A}$, a linkable ring signature scheme satisfies linkability if* $\mathbf{Adv}_{\mathcal{A}}^{link}(\lambda)$ *is negligible.*

Non-Slanderability Non-slanderability is defined in the following game between a challenger $\mathcal{B}$ and the adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}, \mathcal{CO}, \mathcal{SO}$ and the random oracle.

1. $\mathcal{B}$ generates and gives $\mathcal{A}$ the system parameters `pp`.
2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.
3. $\mathcal{A}$ gives $\mathcal{B}$ an event ID e_{ID}, a message m, a set of ℓ public keys $\mathcal{PK}$, where the public key of an insider $pk_\pi \in \mathcal{PK}$ has not been queried to $\mathcal{CO}$ or has not been included as the insider public key of any query to $\mathcal{SO}$. $\mathcal{B}$ uses the private key sk_π corresponding to pk_π to run $\texttt{Sign}(\mathcal{PK}, m, sk_\pi, e_{ID})$ and produces a signature σ' to $\mathcal{A}$.
4. $\mathcal{A}$ queries oracles with arbitrary interleaving. Except pk_π cannot be queried to $\mathcal{CO}$, or included as the insider public key of any query to $\mathcal{SO}$. In particular, $\mathcal{A}$ is allowed to corrupt any public key except for pk_π.
5. $\mathcal{A}$ delivers a set of ℓ^* public keys $\mathcal{PK}^*$, a message m^*, and a signature $\sigma^* \neq \sigma'$.

$\mathcal{A}$ wins the game if

1. $\texttt{Verify}(\mathcal{PK}^*, m^*, \sigma^*, e_{ID}) = 1$;
2. σ^* is not an output of $\mathcal{SO}$;
3. all of the public keys in $\mathcal{PK}^*, \mathcal{PK}$ are query outputs of $\mathcal{JO}$;
4. pk_π has not been queried ot $\mathcal{CO}$; and
5. $\texttt{Link}(\sigma^*, \sigma') = 0$ (i.e., two signatures are linked).

We denote by

$$\mathbf{Adv}_{\mathcal{A}}^{NS}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}].$$

Definition 10 (Non-Slanderability). *For any polynomial time adversary $\mathcal{A}$, a linkable ring signature satisfied non-slanderability if* $\mathbf{Adv}_{\mathcal{A}}^{NS}(\lambda)$ *is negligible.*

3 Post-Quantum (Linkable) Ring Signature

In this section, we describe our construction of DualRing-PRF, a post-quantum ring signature from the Legendre/power residue PRFs (see Section 2.1).

3.1 DualRing-PRF

We extends the DualRing framework [43] to incorporate the Legendre and Power Residue PRFs based post-quantum signature scheme called LegRoast [8]. The setup of our protocol is shown in Algorithm 1. Algorithm 2 shows the steps for key generation. We split the sign algorithm to three parts, as shown in Algorithm 3, 4, 5. The verification algorithm is shown in Algorithm 6 and 7. To improve overall readability in signature generation and verification algorithms, we distinguish the steps related to the application of the Dual-Ring compiler [43] by highlighting them in blue. Steps involving the application of cut-and-choose are highlighted in red, while steps from the original LegRoast protocol are presented in the default text color.

Protocol Overview. Let us denote ℓ as the size of the ring, τ as the number of parallel executions, M as the number of cut-and-choose instances, and N as the number of virtual parties. For simplicity, we provide an overview of the interactive version of our protocol, which can be rendered non-interactive by applying the Fiat-Shamir heuristic [20].

In the first phase, the signer S_π for $\pi \in [\ell]$ begins by sampling $\ell - 1$ random seeds $(c_j)_{j\in[\ell]/\{\pi\}}$ for the non-signers, which are used to expand challenge indices $(I_j^{(b)})_{b\in[B]}$. For each parallel execution, the signer secretly shares the witness K, randomness $(r^{(b)})_{b\in[B]}$, multiplication triples a, b, c such that $c = a \times b$, and masks $(\texttt{mask}_k)_{k\in[M]}$ to N virtual parties. All parties record the shares in their view and commit to them. Next, the signer constructs the commitment ring by computing residuosity symbols $T^{(b)} = \mathcal{L}_0^t(r^{(b)}) - \sum_{j\in[\ell]/\{\pi\}} pk_{I_j^{(b)}}$. The signer sends virtual parties' commitments and residuosity symbols to the verifier. The verifier replies with a short challenge seed h_1 and the signer closes the challenge ring by computing $c_\pi = h_1[:16] \oplus \bigoplus_{j\in[\ell]/\pi} c_j$.[6] The signer then expands c_π to obtain the challenge indices $(I_b^{(\pi)})_{b\in[B]}$.

Subsequently, the signer computes the share adjustments, denoted as $\Delta I_{k,j}^{(b)}$, for all ring members $(I_j^{(b)})_{b\in[B]}$ and masks $(\texttt{mask}_k)_{k\in[M]}$. In the context of the addition secret sharing, the share adjustments are computed as $\Delta I_{k,j}^{(b)} = I_j^{(b)} - \sum_{i=1}^{N}[\texttt{mask}_k]_i$. The signer commits to share adjustments and accumulates *permuted* commitments using Merkle-tree. The accumulator (a.k.a. Merkle root) is then sent to the verifier. The verifier randomly selects $M-1$ masks to open for each parallel computation, while the signer proceeds to compute the responses $o_b = (K + I_\pi^{(b)})r_b$ for all $b \in [B]$. They engage in the MPC-in-the-head protocol for sacrificing based verification [3] of $E = 0$, where

$$E = K \cdot \sum_{b=1}^{B} \lambda^{(b)} r^{(b)} + \sum_{b=1}^{B} \lambda^{(b)}(o^{(b)} - I_\pi^{(b)} r^{(b)}). \tag{2}$$

Once the prover sends the commitment of N views generated during the execution of MPC-in-the-head, the verifier randomly selects $N-1$ parties to corrupt. The prover sends all shares used by the corrupted parties, $M-1$ opened masks $\texttt{mask}_k$, along with the offset $(\Delta I_{\bar{k},j}^{(b)})_{b\in[B]}$ (where $\bar{k} \in [M]$ is the index of the unopened mask) and the accumulator proof, responses $(o_b)_{b\in[B]}$, and all ℓ challenge seeds c_j to the verifier, who then checks

1. if commitments are consistent with the opened shares;

[6] Slash the first 16 bytes of $h_1$to XOR with $\{c_j\}_{j\in[\ell]/\pi}$.

2. if opened views are consistent with each other by executing the MPC-in-the-head honestly with the opened shares;
3. if recomputed share adjustments from $I_j^{(b)}$ and $\mathtt{mask}_k$ are consistent with the accumulator for $M-1$ revealed masks;
4. if the accumulator proof of $\Delta I_{\bar{k},j}^{(b)}$ is valid.
5. if $T^{(b)} = \mathcal{L}_0^t(o^{(b)}) - \sum_{j\in[\ell]/\{\pi\}} pk_{I_j^{(b)}}$.

The formal security proofs including unforgeability and anonymity of DualRing-PRF are shown in Appendix B.

Algorithm 1: DualRing-PRF Setup

```
1  $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
2      On input a security parameter $\lambda$, computes the following public
       parameters:
3          - A large odd prime $p$,
4          - An integer $t \geq 2$ such that $t|(p-1)$,
5          - An integer $L$ (the length of the public key),
6          - A list $\mathcal{I} = (I_1, \ldots, I_L)$ where $I_l$ are randomly chosen from $\mathbb{F}_p$ for all
           $l \in [L]$,
7          - An integer $B \leq L$ (the number of residue symbols that will be
           checked at random),
8          - An integer $N$ (the number of parties),
9          - An integer $\tau$ (the number of parallel executions).
10         - An integer $M$ (the number of total cut-and-choose instances)
11         - Hash functions: $\mathcal{H}_a : \{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$ for $a \in \{\mathtt{sd}, \mathtt{mask}, \Delta, 1, 2, 3, 4\}$
12         - Parameter $\mathtt{acc}_{\mathtt{pp}}$ generated by executing $\mathtt{Acc.Gen}(1^\lambda)$.
13         - A PRG function $\mathtt{Expand}$.
14     Output $\mathtt{pp} = (p, t, L, \mathcal{I}, B, N, \tau, M, \mathcal{H}_{\mathtt{sd}}, \mathcal{H}_{\mathtt{mask}}, \mathcal{H}_\Delta, \mathcal{H}_1,$
       $\mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathtt{acc}_{\mathtt{pp}}, \mathtt{Expand})$.
```

Algorithm 2: DualRing-PRF Key Generation

```
1  $(sk, pk) \leftarrow \mathtt{KGen}(\mathtt{pp})$
2      On input the public parameters $\mathtt{pp}$, computes secret key and public key as
       follows:
3          - Randomly select $K$ from $\mathbb{F}_p^*$ and set $sk = K$,
4          - Compute $pk = \mathcal{L}_K^t(\mathcal{I})$.
5      Output $(sk, pk)$.
```

Algorithm 3: DualRing-PRF Sign - Part I

```
1  $(\sigma) \leftarrow \mathtt{Sign}(\mathcal{PK}, m, sk_\pi)$
2      **Phase 1. Commit to mask, witness, randomness and triples**
3          Pick a random salt $\mathtt{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.
4          Pick random challenges for non-signers $c_j \xleftarrow{\$} \{0,1\}^\lambda$ for $j \in [\ell]/\{\pi\}$.
5          for $e \in [\tau]$ do
6              Sample mask root seed $\mathtt{sd}_e^{(\mathrm{MASK})} \xleftarrow{\$} \{0,1\}^\lambda$, and expand $\mathtt{sd}_e^{(\mathrm{MASK})}$
                to obtain $(\mathtt{sd}_{e,1}^{(\mathrm{MASK})}, \ldots, \mathtt{sd}_{e,M}^{(\mathrm{MASK})})$ using a tree-based PRG.
7              for $k \in [M]$ do
8                  Compute mask shares $([\mathtt{mask}_{e,k}]_i)_{i \in [N]}$ from $\mathtt{sd}_{e,k}^{(\mathrm{MASK})}$ with
                    tree-based PRG.
9              Sample MPC root seed $\mathtt{sd}_e^{(\mathrm{MPC})} \xleftarrow{\$} \{0,1\}^\lambda$ and compute
                $(\mathtt{sd}_{e,i}^{(\mathrm{MPC})}, \ldots, \mathtt{sd}_{e,N}^{(\mathrm{MPC})})$ from $\mathtt{sd}_e^{(\mathrm{MPC})}$ with tree-based PRG.
10             for $i \in [N]$ do
11                 Sample MPC shares by expanding the seeds
                    $([K_e]_i, [r_e^{(1)}]_i, \ldots, [r_e^{(B)}]_i, [a_e]_i, [b_e]_i, [c_e]_i) \leftarrow \mathtt{Expand}(sd_{e,i}^{(\mathrm{MPC})})$.
12                 Commit to MPC seeds $C_{e,i}^{(\mathrm{MPC})} \leftarrow \mathcal{H}_{\mathtt{sd}}(\mathtt{salt}, e, i, \mathtt{sd}_{e,i}^{(\mathrm{MPC})})$.
13                 for $k \in [M]$ do
14                     Commit to mask shares
                        $C_{e,k,i}^{(\mathrm{MASK})} \leftarrow \mathcal{H}_{\mathtt{mask}}(\mathtt{salt}, e, k, i, [\mathtt{mask}_{e,k}]_i)$
15             Compute key adjustment $\Delta K_e \leftarrow K - \sum_{i=1}^N [K_e]_i$ and set
                $[K_e]_1 \leftarrow [K_e]_1 + \Delta K_e$.
16             Compute triple adjustment
                $\Delta c_e = (\sum_{i=1}^N [a_e]_i) \cdot (\sum_{i=1}^N [b_e]_i) - (\sum_{i=1}^N [c_e]_i)$ and set
                $[c_e]_1 = [c_e]_1 + \Delta c_e$.
17             for $j \in [\ell]/\{\pi\}$ do
18                 Expand challenge $c_j$, $(I_{e,j}^{(1)}, \ldots, I_{e,j}^{(B)}) \leftarrow \mathtt{Expand}(c_j)$.
19             for $b \in [B]$ do
20                 Compute residuosity symbols $T_e^{(b)} = \mathcal{L}_0^t(r_e^{(b)})$
                    $- \sum_{j \in [\ell]/\{\pi\}} pk_{I_{e,j}^{(b)}}$, where $r_e^{(b)} = \sum_{i=1}^N [r_e^{(b)}]_i$.
21         $\sigma_1 \leftarrow (((C_{e,k,i}^{(\mathrm{MASK})})_{k \in [M]}, C_{e,i}^{(\mathrm{MPC})})_{i \in [N]}, (T_e^{(b)})_{b \in [B]}, \Delta K_e, \Delta c_e)_{e \in [\tau]}$.
```

Algorithm 4: DualRing-PRF Sign - Part II

```
1  Phase 2. Compute mutual challenge and challenge offsets
2      Compute $h_1 \leftarrow \mathcal{H}_1(\texttt{salt}, m, \mathcal{PK}, \sigma_1)$.
3      Compute real signer's challenge $c_\pi = h_1[1,128] \oplus \bigoplus_{j \in [\ell]/\{\pi\}} c_j$. ($h_1[1,128]$
         denotes the use of the first 128 bits of $h_1$.)
4      Expand $c_\pi$, $(I_{e,\pi}^{(1)}, \ldots, I_{e,\pi}^{(B)}) \leftarrow \texttt{Expand}(c_\pi)$.
5      for $e \in [\tau]$ do
6          for $k \in [M]$ do
7              Compute $\texttt{mask}_{e,k} = \sum_{i=1}^{N} [\texttt{mask}_{e,k}]_i$.
8              for $j \in [\ell]$ do
9                  for $b \in [B]$ do
10                     Compute offset $\Delta I_{e,k,j}^{(b)} \leftarrow I_{e,j}^{(b)} - \texttt{mask}_{e,k}$.
11                 Commit to offsets $C_{e,k,j}^{(\Delta)} = \mathcal{H}_\Delta(\texttt{salt}, e, k, (\Delta I_{e,k,j}^{(1)}, \ldots, I_{e,k,j}^{(B)}))$.
12             Randomly samples a permutation $\phi_{e,k}$.
13             Accumulate the permuted commitments using Merkle tree:
                 $\texttt{acc}_{e,k} \leftarrow \texttt{Acc.Eval}(\texttt{pp}, C_{e,k,\phi_{e,k}(1)}^{(\Delta)}, \ldots, C_{e,k,\phi_{e,k}(\ell)}^{(\Delta)})$
14         Combine $M$ accumulators using a hash function
             $\texttt{acc}_e = \mathcal{H}_{\texttt{acc}}(\texttt{salt}, e, (\texttt{acc}_{e,1}, \ldots, \texttt{acc}_{e,M}))$.
15     Set $\sigma_2 \leftarrow (\texttt{acc}_e)_{e \in [\tau]}$.
16 Phase 3. Compute opened executions for cut-and-choose
17     Compute $h_2 \leftarrow \mathcal{H}_2(h_1, \sigma_2)$.
18     Expand $h_2$: $(\bar{k}_e)_{e \in [\tau]} \leftarrow \texttt{Expand}(h_2)$ such that $\bar{k}_e \in [M]$.
19 Phase 4. Compute output values
20     for $e \in [\tau]$ do
21         for $b \in [B]$ do
22             $o_e^{(b)} = (K + I_{e,\pi}^{(b)}) r_e^{(b)}$.
23     Set $\sigma_3 \leftarrow (o_e^{(1)}, \ldots, o_e^{(B)})_{e \in [\tau]}$.
24 Phase 5. Compute challenges for sacrificing based verification
25     Compute $h_3 \leftarrow \mathcal{H}_3(h_2, \sigma_3)$.
26     Expand hash $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)})_{e \in [\tau]} \leftarrow \texttt{Expand}(h_3)$, where $\epsilon_e, \lambda_e^{(b)} \in \mathbb{Z}_p$.
```

Algorithm 5: DualRing-PRF Sign - Part III

```
1  Phase 6. Commit to views of sacrificing protocol
2      for $e \in [\tau]$ do
3          for $i \in [N]$ do
4              Compute shares $[\alpha_e]_i \leftarrow [a_e]_i + \epsilon_e[K_e]_i$ and
                 $[\beta_e]_i \leftarrow [b_e]_i + \sum_{b=1}^{B} \lambda_e^{(b)}[r_e]_i^{(b)}$.
5          Compute $\alpha_e \leftarrow \sum_{i=1}^{N}[\alpha_e]_i$ and $\beta_e \leftarrow \sum_{i=1}^{N}[\beta_e]_i$.
6          for $i \in [N]$ do
7              for $b \in [B]$ do
8                  if $i = 1$ then
9                      Define the challenge share $[I_e^{(b)}]_i \leftarrow [\texttt{mask}_{e,\bar{k}_e}]_i + \Delta I_{e,\bar{k}_e,\pi}^{(b)}$.
10                 else
11                     Define the challenge share $[I_e^{(b)}]_i \leftarrow [\texttt{mask}_{e,\bar{k}_e}]_i$.
12             Compute $[z_e]_i \leftarrow \sum_{b=1}^{B} -\lambda_e^{(b)}[r_e^{(b)}]_i I_{e,\bar{k}_e,pi}^{(b)}$ and
                 $[z_e]_i' \leftarrow \sum_{b=1}^{B} -\lambda_e^{(b)}[r_e^{(b)}]_i[I_e^{(b)}]_i$.
13             Compute $z_e = \sum_{i=1}^{N}[z_e]_i$ and $\Delta z_e = z_e - \sum_{i=1}^{N}[z_e]_i'$.
14             if $i = 1$ then
15                 $[z_e]_i' = [z_e]_i' + \Delta z_e + \sum_{b=1}^{B} \lambda_e^{(b)} o_e^{(b)}$.
16             Compute check share values
                 $[\gamma_e]_i \leftarrow \alpha_e[b_e]_i + \beta_e[a_e]_i - [c_e]_i + \epsilon_e[z_e]_i'$.
17     Set $\sigma_4 \leftarrow (\alpha_e, \beta_e, ([\alpha_e]_i, [\beta_e]_i, [\gamma_e]_i)_{i\in[N]})_{e\in[\tau]}$.
18 Phase 7. Challenge on sacrificing protocol
19     Compute challenge $h_4 \leftarrow \mathcal{H}_4(h_3, \sigma_4)$.
20     Expand challenge $(\bar{i}_e)_{e\in[\tau]} \leftarrow \texttt{Expand}(h_4)$, where $\bar{i}_e \in [N]$.
21 Phase 8. Opening executions for cut-and-choose and MPC views
22     for $e \in \tau$ do
23         Define $\texttt{seeds}_e^{(\mathrm{MPC})}$ be the $\log_2 N$ seeds in tree to compute $\texttt{sd}_{e,i}^{(\mathrm{MPC})}$ for
             $i \neq \bar{i}_e$.
24         Define $\texttt{seeds}_e^{(\mathrm{MASK})}$ be the $\log_2 M$ seeds in tree to compute $\texttt{sd}_{e,k}^{(\mathrm{MASK})}$
             for $k \neq \bar{k}_e$.
25         Define $\texttt{masks}_{e,\bar{k}_e}$ be the $\log_2 N$ seeds in tree to compute $[\texttt{mask}_{e,\bar{k}_e}]_i$ for
             $i \neq \bar{i}_e$.
26         Define $\texttt{perms}_e$ be the $\log_2 M$ seeds to compute permutation initiation
             vector $\phi_{e,k}$ for $k \neq \bar{k}_e$.
27         Define $\texttt{auth}_{e,\bar{k}_e}$ be the Merkle tree proof computed from
             $\texttt{Acc.Proof}(\texttt{pp}, \texttt{acc}_{e,\bar{k}_e}, C_{e,\bar{k}_e,\phi_{e,\bar{k}_e}(\pi)}^{(\Delta)})$.
28         Define $\Delta I_{e,\bar{k}_e}^{(b)} \leftarrow \Delta I_{e,\bar{k}_e,\pi}^{(b)}$ for all $b \in [B]$.
29     Set signature $\sigma =$
         $(\texttt{salt}, h_2, h_4, (\texttt{perms}_e, \texttt{seeds}_e^{(\mathrm{MASK})}, \texttt{masks}_{e,\bar{k}_e}, \texttt{auth}_{e,\bar{k}_e}, \texttt{seeds}_e^{(\mathrm{MPC})}, \Delta K_e,$
         $\Delta c_e, \Delta z_e, (o_e^{(b)}, \Delta I_{e,\bar{k}_e}^{(b)})_{b\in[B]}, \alpha_e, \beta_e, C_{e,\bar{i}_e}^{(\mathrm{SD})}, C_{e,\bar{k}_e,\bar{i}_e}^{(\mathrm{MASK})})_{e\in[\tau]}, (c_j)_{j\in[\ell]})$.
```

Algorithm 6: DualRing-PRF Verify - Part I

```
1  $(0/1) \leftarrow \texttt{Verify}(\mathcal{PK}, m, \sigma)$
2    $\sigma =$
       $(\texttt{salt}, h_2, h_4, (\texttt{perms}_e, \texttt{seeds}_e^{(\mathrm{MASK})}, \texttt{masks}_{e,\bar{k}_e}, \texttt{auth}_{e,\bar{k}_e}, \texttt{seeds}_e^{(\mathrm{MPC})}, \Delta K_e,$
       $\Delta c_e, \Delta z_e, (o_e^{(b)}, \Delta I_{e,\bar{k}_e}^{(b)})_{b\in[B]}, \alpha_e, \beta_e, C_{e,\bar{i}_e}^{(\mathrm{SD})}, C_{e,\bar{k}_e,\bar{i}_e}^{(\mathrm{MASK})})_{e\in[\tau]}, (h_j)_{j\in[\ell]})$.
3    for $j \in [\ell]$ do
4      $(I_{e,j}^{(1)}, \ldots, I_{e,j}^{(B)})_{e\in[M]} \leftarrow \texttt{Expand}(c_j)$.
5    Expand $h_2, h_3, h_4$: $(\bar{k}_e)_{e\in[\tau]} \leftarrow \texttt{Expand}(h_2)$;
       $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)})_{e\in\tau} \leftarrow \texttt{Expand}(h_3)$; $(\bar{i}_e)_{e\in\tau} \leftarrow \texttt{Expand}(h_4)$, where
       $h_3 = \mathcal{H}_3(h_2, (o_e^{(1)}, \ldots, o_e^{(B)})_{e\in[\tau]})$.
6    for $e \in [\tau]$ do
7      Use $\texttt{seeds}_e$ to compute $\texttt{sd}_{e,i}^{(\mathrm{MPC})}$ and $\texttt{masks}_{e,\bar{k}_e}$ to compute $[\texttt{mask}_{e,\bar{k}_e}]_i$
         for $i \neq \bar{i}_e$.
8      for $i \in [N]/\{\bar{i}_e\}$ do
9        Sample shares
           $([K_e]_i, [r_e^{(1)}]_i, \ldots, [r_e^{(B)}]_i, [a_e]_i, [b_e]_i, [c_e]_i) \leftarrow \texttt{Expand}(sd_{e,i}^{(\mathrm{MPC})})$.
10       Recompute the commitment $C_{e,i}^{(\mathrm{MPC})} \leftarrow \mathcal{H}_{\texttt{sd}}(\texttt{salt}, e, i, \texttt{sd}_{e,i}^{(\mathrm{MPC})})$.
           and $C_{e,\bar{k}_e,i}^{(\mathrm{MASK})} \leftarrow \mathcal{H}_{\texttt{mask}}(\texttt{salt}, e, \bar{k}_e, i, [\texttt{mask}_{e,\bar{k}_e}]_i)$.
11       if $i = 1$ then
12         Adjust the share $[K_e]_i \leftarrow [K_e]_i + \Delta K_e$ and $[c_e]_i \leftarrow [c_e]_i + \Delta c_e$.
13       for $b \in [B]$ do
14         Set $[I_e^{(b)}]_i \leftarrow [\texttt{mask}_{e,\bar{k}_e}]_i$.
15         if $i = 1$ then
16           Adjust the share $[I_e^{(b)}]_i \leftarrow [\texttt{mask}_{e,\bar{k}_e}]_i + \Delta I_{e,\bar{k}_e}^{(b)}$.
17       Recompute shares $[\alpha_e]_i \leftarrow [a_e]_i + \epsilon_e[K_e]_i$,
           $[\beta_e]_i \leftarrow [b_e]_i + \sum_{b=1}^{B} \lambda_e^{(b)}[r_e^{(b)}]_i$, and
           $[z_e]_i \leftarrow \sum_{b=1}^{B} -\lambda_e^{(b)}[r_e^{(b)}]_i[I_e^{(b)}]_i$.
18       if $i = 1$ then
19         $[z_e]_i \leftarrow [z_e]_i + \sum_{b=1}^{B} \lambda_e^{(b)} o_e^{(b)} + \Delta z_e$.
20       Recompute check value shares
           $[\gamma_e]_i \leftarrow \alpha_e[b_e]_i + \beta_e[a_e]_i - [c_e]_i + \epsilon_e[z_e]_i$.
21       Compute missing shares: $[\alpha_e]_{\bar{i}_e} \leftarrow \alpha_e - \sum_{i\neq\bar{i}_e}[\alpha_e]_i$,
           $[\beta_e]_{\bar{i}_e} \leftarrow \beta_e - \sum_{i\neq\bar{i}_e}[\beta_e]_i$, and $[\gamma_e]_{\bar{i}_e} \leftarrow \alpha_e\beta_e - \sum_{i\neq\bar{i}_e}[\gamma_e]_i$
22     for $b \in [B]$ do
23       $T_e^{(b)} \leftarrow \mathcal{L}_0^t(o_e^{(b)}) - \sum_{j=1}^{\ell} pk_{I_{e,j}^{(b)}}$.
```

Algorithm 7: DualRing-PRF Verify - Part II

```
1  for $e \in [\tau]$ do
2      Use $\texttt{perms}_e$ to compute permutation initialisation vectors $\phi_{e,k}$ and
       $\texttt{seeds}_e^{(\mathrm{MASK})}$ to compute $\texttt{sd}_{e,k}^{(\mathrm{MASK})}$ for $k \neq \bar{k}_e$.
3      for $k \in [M]/\{\bar{k}_e\}$ do
4          Compute all mask shares $([\texttt{mask}_{e,k}]_i)_{i\in[N]}$ using $\texttt{sd}_{e,k}^{(\mathrm{MASK})}$.
5          for $i \in [N]$ do
6              Commit to mask shares $C_{e,k,i}^{(\mathrm{MASK})} \leftarrow \mathcal{H}_{\texttt{mask}}(\texttt{salt}, e, k, i, [\texttt{mask}_{e,k}]_i)$
7          Compute $\texttt{mask}_{e,k} = \sum_{i=1}^{N} [\texttt{mask}_{e,k}]_i)$.
8          for $j \in [\ell]$ do
9              for $b \in [B]$ do
10                 Compute $\Delta I_{e,k,j}^{(b)} = I_{e,j}^{(b)} - \texttt{mask}_{e,k}$.
11             Compute $C_{e,k,j}^{(\Delta)} = \mathcal{H}_\Delta(\texttt{salt}, e, k, (\Delta I_{e,k,j}^{(1)}, \ldots, \Delta I_{e,k,j}^{(B)}))$.
12         Compute $\texttt{acc}_{e,k} \leftarrow \texttt{Acc.Eval}(\texttt{pp}, C_{e,k,\phi_{e,k}(1)}^{(\Delta)}, \ldots, C_{e,k,\phi_{e,k}(\ell)}^{(\Delta)})$
13     Recompute $C_{e,\bar{k}_e}^{(\Delta)} \leftarrow \mathcal{H}_\Delta(\texttt{salt}, e, \bar{k}, (\Delta I_{e,\bar{k}_e}^{(1)}, \ldots, \Delta I_{e,\bar{k}_e}^{(B)}))$.
14     Use $C_{e,\bar{k}_e}^{(\Delta)}$ as the leaf node and $\texttt{auth}_{e,\bar{k}_e}$ as the authentication path to
       recompute $\texttt{acc}_{e,\bar{k}_e}$.
15     Recompute $\texttt{acc}_e = \mathcal{H}_{\texttt{acc}}(\texttt{salt}, e, (acc_{e,1}, \ldots, acc_{e,M}))$.
16 Compute $h_1 = \bigoplus_{j=1}^{\ell} c_j$, and check if
   $h_1 \overset{?}{=} \mathcal{H}_1((\texttt{salt}, m, \mathcal{PK}, ((C_{e,k,i}^{(\mathrm{MASK})})_{k\in[M]}, C_{e,i}^{(\mathrm{MPC})})_{i\in[N]}, (T_e^{(b)})_{b\in[B]},$
   $\Delta K_e, \Delta c_e)_{e\in[\tau]})$.
17 Check if $h_2 \overset{?}{=} \mathcal{H}_2(h_1, (\texttt{acc}_e)_{e\in[M]})$.
18 Check if $h_4 \overset{?}{=} \mathcal{H}_4(h_3, (\alpha_e, \beta_e, (\alpha_{e,i}, \beta_{e,i}, \gamma_{e,i})_{i\in[N]})_{e\in[\tau]})$.
19 Output accept if and only if all checks pass.
```

3.2 Linkable DualRing-PRF

We optimise the technique for DualRing-PRF to significantly reduce the communication overhead in the linkable variant of DualRing-PRF (referred to as DualRing$_L$-PRF). Recall that our construction instantiates the NP relation with Legendre/power residue PRFs, as introduced in Section 1.2.

Let e_{ID} denote the event ID and $h_{ID} = \mathcal{H}(e_{ID})$ denote the digest of the event ID. We compute the linking tag as follows:

1. We expand h_{ID} to obtain a unique set of L elements $\mathcal{I}\prime = \{I'_1, \ldots, I'_L\}$.
2. The linking tag is computed as $\texttt{tag} = \mathcal{L}^t_K(I'_1), \ldots, \mathcal{L}^t_K(I'_L)$, where K is the user's secret key.

Note that the length of the linking tag is the same as the length of a public key.

Our main idea is to let the signer compute a response $o'_b = (K + I'_b)r'_b$ along with the original $o_b = (K + I^{(b)}_\pi)r_b$ using the same secret key K. Then, we apply the MPC-in-the-head based zero-knowledge proof to show that both o'_b and o_b are correctly computed for all $b \in [B]$. Naively, the approach results in a communication overhead that is twice of the original ring signature scheme, since the signer runs the zero-knowledge proof twice (one for proving o_b and one for o'_b). We exploit the idea of random linear combination to optimise the signature size for our linkable ring signature. That is, instead of proving two responses separately, we prove the following equation using the MPC-in-the-head based zero-knowledge proof:

$$0 = \sum_{b=1}^{B} \Big(\lambda_b(o_b - (K + I^{(b)}_\pi)r_b) + \lambda'_b(o'_b - (K + I'_b)r'_b) \Big) .$$

Note that for fixed o_b and o'_b, if $o_b \neq (K + I^{(b)}_\pi)r_b$, then $E = \sum_{b=1}^{B}(o_b - (K + I^{(b)}_\pi)r_b)$ where $E = 0$ has error probability $1/p$. Similarly, let $E' = \sum_{b=1}^{B}(o'_b - (K + I'_b)r_b)$, then $E' = 0$ has probability $1/p$ if $o'_b \neq (K + I'_b)r'_b$. For uniformly chosen $(\lambda_b, \lambda'_b)_{b\in[B]}$ that are independent with each other, the probability that $E + E' = 0$ is at most $1/p$.

We construct a linkable DualRing-PRF (DualRing$_L$-PRF) which can effectively determine whether two signatures are generated by the same signer, with respect to an event ID. The setup phase of DualRing$_L$-PRF remains the same as shown in Algorithm 1, except that we define an additional hash function $\mathcal{H}_{ID} : \{0,1\}^* \to \{0,1\}^\lambda$, which on input an event ID, outputs a short digest. The key generation remains unchanged (refer to Algorithm 2). The process of `Link` algorithm is shown as follows:

On input two valid signatures σ_1 and σ_2 with respect to $(\mathcal{PK}_1, m_1)$ and $(\mathcal{PK}_2, m_2)$, parse $\sigma_1 = (\ldots, \texttt{tag}_1)$ and $\sigma_2 = (\ldots, \texttt{tag}_2)$. The algorithm outputs 0 (linked) if $\texttt{tag}_1 = \texttt{tag}_2$. Otherwise, output 1 (unlinked).

We present the signing and verification of the linkable DualRing-PRF in Algorithm 8 and 9, respectively. The formal security proofs including linkability and non-slanderability of DualRing$_L$-PRF are shown in Appendix C.

Algorithm 8: Linkable DualRing-PRF Sign

```
1  $(\sigma) \leftarrow \texttt{Sign}(\mathcal{PK}, m, sk_\pi, e_{ID})$
2      Compute $h_{ID} = \mathcal{H}_{ID}(e_{ID})$ and $\mathcal{I}' = (I'_1, \ldots, I'_L) \leftarrow \texttt{Expand}(h_{ID})$, where
         $I_l \in \mathbb{F}_p$ for $l \in [L]$.
3      Compute the linking tag $\texttt{tag} = \mathcal{L}^t_K(\mathcal{I}')$.
4      Proceed with line 4 - 11 as shown in Algorithm 3.
5      for $e \in [\tau]$ AND $i \in [N]$ do
6          Replace line 12 in Algorithm 3 with
             $([K_e]_i, [r_e^{(1)}]_i, \ldots, [r_e^{(B)}]_i, [r_e^{(1)\prime}]_i, \ldots, [r_e^{(B)\prime}]_i, [a_e]_i,$
             $[b_e]_i, [c_e]_i) \leftarrow \texttt{Expand}(\texttt{sd}^{(MPC)}_{e,i})$.
7          Note that we require $\sum_{i=1}^N r_e^{(b)} \neq \sum_{i=1}^N r_e^{(b)\prime}$.
8      Continue with line 13 - 21 as shown in Algorithm 3, and add an additional
         step as follows:
9      for $b \in [B]$ do
10         $T_e^{(b)\prime} = \mathcal{L}^t_0(r_e^{(b)\prime})$ where $r_e^{(b)\prime} = \sum_{i=1}^N [r_e^{(b)\prime}]_i$.
11     Replace line 22
         with:$\sigma_1 \leftarrow ((C^{(\mathrm{MASK})}_{e,k,i})_{k\in[M]}, C^{(\mathrm{MPC})}_{e,i})_{i\in[N]}, (T_e^{(b)}, T_e^{(b)\prime})_{b\in[B]}, \Delta K_e,$
         $\Delta c_e)_{e\in[\tau]}$.
12     Continue with line 1 - 3 in Algorithm 4 and add an additional step after
         line 3 as follows: $\mathcal{I}'$: $(I_e^{(1)'}, \ldots, I_e^{(B)\prime})_{e\in[\tau]} \leftarrow \texttt{Expand}(h_1)$.
13     Continue with line 4 -22 as shown in Algorithm 4 and add an additional
         step after line 22 as follows:
14     for $e \in [\tau]$ AND $b \in [B]$ do
15         Compute additional response $o'_b = (K + I_e^{(b)\prime}) r_e^{(b)\prime}$.
16     Replace line 23 with: $\sigma_3 \leftarrow (o_e^{(1)}, \ldots, o_e^{(B)}, o_e^{(1)\prime}, \ldots, o_e^{(B)\prime})_{e\in[\tau]}$.
17     Continue with line 24 - 25 as shown in Algorithm 4.
18     Replace line 26 in Algorithm 4 with
         $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)}, \lambda_e^{(1)\prime}, \ldots, \lambda_e^{(B)\prime}) \leftarrow \texttt{Expand}(h_3)$.
19     Replace line 2 - 6 in Algorithm 5 with:
20     for $e \in [\tau]$ do
21         for $i \in [N]$ do
22             Compute shares $[\alpha_e]_i \leftarrow [a_e]_i + \epsilon_e [K]_i$ and
                 $[\beta_e]_i \leftarrow [b_e]_i + \sum_{b=1}^B \lambda_e^{(b)} [r_e^{(b)}]_i + \lambda_e^{(b)\prime} [r_e^{(b)\prime}]_i$.
23             Continue with line 6 - 11.
24             Replace line 12 - 13 as
                 $[z_e]_i \leftarrow \sum_{b=1}^B -(\lambda_e^{(b)} [r_e^{(b)}]_i I^{(b)}_{e,\bar{k}_e,\pi} + \lambda_e^{(b)\prime} [r_e^{(b)\prime}]_i I_e^{(b)\prime})$ and
                 $[z_e]'_i \leftarrow \sum_{b=1}^B -(\lambda_e^{(b)} [r_e^{(b)}]_i [I_e^{(b)}]_i + \lambda_e^{(b)\prime} [r_e^{(b)\prime}]_i I_e^{(b)\prime})$
25             Continue with line 13 and replace line 14 - 15:
26             if $i = 1$ then
27                 $[z_e]_i = [z_e]_i + \sum_{b=1}^B \lambda_e^{(b)} o_e^{(b)} + \lambda_e^{(b)\prime} o_e^{(b)\prime}$.
28     Continue with line 16 - 28 and replace line 29 as
         $\sigma = (\texttt{salt}, h_2, h_4, (\texttt{perms}_e, \texttt{seeds}_e^{(\mathrm{MASK})}, \texttt{masks}_{e,\bar{k}_e}, \texttt{auth}_{e,\bar{k}_e},$
         $\texttt{seeds}_e^{(\mathrm{MPC})}, \Delta K_e, \Delta c_e, \Delta z_e, (o_e^{(b)}, o_e^{(b)\prime}, \Delta I^{(b)}_{e,\bar{k}_e})_{b\in[B]}, \alpha_e, \beta_e, C^{(\mathrm{SD})}_{e,\bar{i}_e},$
         $C^{(\mathrm{MASK})}_{e,\bar{k}_e,\bar{i}_e})_{e\in[\tau]}, (c_j)_{j\in[\ell]})$ and output the linking tag $\texttt{tag}$.
```

Algorithm 9: Linkable DualRing-PRF Verify

```
1  $(0/1) \leftarrow \texttt{Verify}(\mathcal{PK}, m, \sigma, e_{ID})$
2      Compute $h_{ID} = \mathcal{H}_{ID}(e_{ID})$ and $(I'_1, \ldots, I'_L) \leftarrow \texttt{Expand}(h_{ID})$.
3      Proceed with line 1 - 3 in Algorithm 6.
4      Add the following step after line 3: Compute $h_1 \leftarrow \bigoplus_{j=1}^{\ell} c_j$ and expand
         hash $(I_e^{(1)\prime}, \ldots, I_e^{(B)}\prime)_{e\in[\tau]} \leftarrow \texttt{Expand}(h_1)$.
5      Replace line 4 with: $(\bar{k}_e)_{e\in[\tau]} \leftarrow \texttt{Expand}(h_2)$;
         $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)}, \lambda_e^{(1)\prime}, \ldots, \lambda_e^{(B)\prime})_{e\in\tau} \leftarrow \texttt{Expand}(h_3)$;
         $(\bar{i}_e)_{e\in\tau} \leftarrow \texttt{Expand}(h_4)$, where
         $h_3 = \mathcal{H}_3(h_2, (o_e^{(1)}, \ldots, o_e^{(B)}, o_e^{(1)\prime}, \ldots, o_e^{(B)\prime})_{e\in[\tau]})$.
6      Continue with line 5 - 7, replace line 8 with
         $([K_e]_i, [r_e^{(1)}]_i, \ldots, [r_e^{(B)}]_i, [r_e^{(1)\prime}]_i, \ldots, [r_e^{(B)\prime}]_i, [a_e]_i, [b_e]_i,$
         $[c_e]_i) \leftarrow \texttt{Expand}(\texttt{sd}_{e,i}^{(MPC)})$.
7      Continue with line 9 - 15, replace line 16 - 18 with:
8      $[\alpha_e]_i \leftarrow [a_e]_i + \epsilon_e[K]_i$, $[\beta_e]_i \leftarrow [b_e]_i + \sum_{b=1}^{B} \lambda_e^{(b)}[r_e^{(b)}]_i + \lambda_e^{(b)\prime}[r_e^{(b)\prime}]_i$, and
         $[z_e]_i \leftarrow \sum_{b=1}^{B} -(\lambda_e^{(b)}[r_e^{(b)}]_i[I_e^{(b)}]_i + \lambda_e^{(b)\prime}[r_e^{(b)\prime}]_i I_e^{(b)\prime})$.
9      if $i = 1$ then
10         $[z_e]_i = [z_e]_i + \sum_{b=1}^{B} \lambda_e^{(b)} o_e^{(b)} + \lambda_e^{(b)\prime} o_e^{(b)\prime}$.
11     Continue with Line 19 - 22, add the following step after line 22:
         $T_e^{(b)\prime} = \mathcal{L}_0^t(o_e^{(b)\prime}) - \texttt{tag}_{I_e^{(b)\prime}}$.
12     Continue with line 23 in Algorithm 6 and 1 - 14, remove line 15, and
         replace line 16 with
         $h_1 \overset{?}{=} \mathcal{H}_1((\texttt{salt}, m, \mathcal{PK}, ((C_{e,k,i}^{(\text{MASK})})_{k\in[M]}, C_{e,i}^{(\text{MPC})})_{i\in[N]}, (T_e^{(b)}, T_e^{(b)\prime})_{b\in[B]},$
         $\Delta K_e, \Delta c_e)_{e\in[\tau]})$.
13     Continue with line 17 - 19.
```

4 Performance Evaluation

We implemented the DualRing-PRF in Python and benchmark the performance. Our experiments are run on a Macbook Pro with Apple M1 Max chip and 32GB memory. We provide multiple choice of parameters that creates a trade-off between signature size and signing speed.

4.1 Choice of Parameters

For both DualRing-PRF and linkable DualRing-PRF (DualRing$_L$-PRF), we choose the prime p to have size approximately 2^λ, where λ is the security parameter. Specifically, when targeting 128-bit security, we use the Mersenne prime $p = 2^{127}-1$. For power residue PRFs, we follow the parameter choice in LegRoast [8], which uses $t = 254$ such that the power residue symbol can be effectively represented by a single byte.

We choose $L = 2^{15}$ for Legendre PRF and $L = 2^{12}$ for power-residue PRF as in LegRoast [8], which results in public key size of 4KB. To ensure that β-approximate PRF relation (Definition 3) is as hard as the original relation (Definition 2), we choose $\beta = 0.449$ for the Legendre PRF and $\beta = 0.967$ for the power-residue PRF (same as in LegRoast).

The unforgeability proof shows that the best attacking strategy is to query $\mathcal{H}_1$ on many inputs and choose the query for which $T_e^{(b)} = \mathcal{L}_0^t(o_e^{(b)}) - \sum_{j=1}^{\ell} pk_{I_{e,j}^{(b)}}$ hold for the most executions. Suppose this is the case for τ' out of τ executions. Then, for the remaining $\tau - \tau'$ executions, the adversary computes one dishonest share adjustments for $I_{e,\pi}^{(b)}$ and queries $\mathcal{H}_2$, in the hope of getting an output $(\bar{k}_e)_{e\in[\tau]}$ that asks him to open all honestly computed share adjustments. This succeeds with probability $M^{-(\tau-\tau')}$. Let us assume that this is the case for τ'' executions. The adversary then makes one of the virtual parties cheat in the MPC protocol in each of the $\tau - \tau' - \tau''$ executions, and hope that $(\bar{i}_e)_{e\in[\tau]}$ opens non-cheating parties. This succeeds with probability $N^{-(\tau-\tau'-\tau'')}$.

Therefore, to achieve λ bits of security, we take parameters B, M, N and τ such that

$$\begin{aligned} 2^\lambda \leq \min_{\tau'\in\{0,\ldots,\tau\},\tau''\in\{0,\ldots,\tau-\tau'\}} & (\Pr[\mathfrak{B}(\tau,(1-\beta)^B) \geq \tau']^{-1} \\ & + \Pr[\mathfrak{B}(\tau-\tau', 1/M) \geq \tau'']^{-1} + N^{\tau-\tau'-\tau''}) \end{aligned} \quad (3)$$

To choose parameters, we fix an integer for M and varying N, then we compute values of B and τ that satisfies the inequality 3. Note that the choice of N controls a trade-off between algorithm running time and signature size. Namely, if N is large, then the soundness error for MPC-in-the-head is small, hence requires less number of repetitions, which results in a smaller signature size. We show the parameter choice and corresponding signature size in Table 2. We choose the same parameters for linkable DualRing-PRF, which results in linking tag of the same size as the public key.[7] We refer the readers to Appendix D for a detailed signature size analysis.

We illustrate the signing time of DualRing-PRF based on Legendre (left) and power-residue PRFs (right) in Figure 1.[8] The evaluation shows that, DualRing-PRF based on power-residue PRF with fast parameters result in similar signing time as DualRing-LB (i.e., approximately 25 seconds for a ring of size 256).

	N	B	τ	M	$n = 2^4$	2^6	2^8	2^{10}	`tag`	$\lvert\pi_L\rvert$
L-Fast	16	10	74	2^7	67	72	80	97		12
L-Medium	64	10	58	2^8	58	62	69	84	4	10
L-Slow	256	11	46	2^9	53	57	63	77		9
PR-Fast	16	5	59	2^7	44	49	55	71		5
PR-Medium	64	6	43	2^8	37	41	47	61	4	4
PR-Slow	256	7	34	2^9	34	37	42	56		4

Table 2. Parameter Choices for (Linkable) DualRing-PRF. Sizes are presented by KB. N = # of MPC parties, B = # of checked PRF symbols, τ = # of protocol repetition, M = cut-and-choose parameter, n = ring size, "`tag`" = the size of linking tag; $|\pi_L|$ = the additional proof size for linkable DualRing-PRF.

Despite the fact that DualRing-PRF and linkable DualRing-PRF are not comparable with lattice-based ones in terms of signature sizes, our scheme relies on different post-quantum security assumptions. As also evident from NIST's selection of SPHINCS+ [5] for standardisation despite its significant efficiency disadvantages compared to lattice-based alternatives. As such, exploring post-quantum cryptographic primitives based on a range of security assumptions becomes critically important.

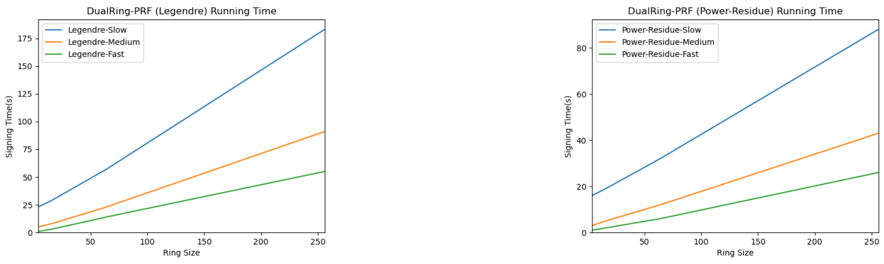

Fig. 1. Signing Time for DualRing-PRF based on Power-Residue and Legendre PRFs.

[7] L controls the trade-off between signature size and public key size. Smaller L results in smaller public key but larger signature. We refer the readers to [8, Remark 1] for more details.

[8] The verification time of DualRing-PRF is approximately equal to (though slightly smaller than) the signing time, and therefore is not included in the evaluation.

5 Conclusion

In this work, we present symmetric-key primitives based post-quantum (linkable) ring signatures that have the most competitive signature sizes among the schemes based on the similar security assumption for small and medium rings. Our ring signature is 1.4 times smaller than the state-of-the-art symmetric-key based ring signature [22]. Comparing with one-time traceable ring signature [38], our construction not only supports many-time signing, but also achieves significantly smaller signature sizes when ring size exceeds 16. This advantage becomes more pronounced as the ring size increases.

5.1 Future Works

The significant performance gap between lattice-based and symmetric-key based ring and linkable ring signatures persists. Exploring methods to narrow this gap could be an interesting topic for further research.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: ASIACRYPT. pp. 415–432. Springer (2002)
2. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: logarithmic-size, no setup—from standard assumptions. In: EUROCRYPT. pp. 281–311. Springer (2019)
3. Baum, C., Nof, A.: Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In: PKC. pp. 495–526. Springer (2020)
4. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: TCC. pp. 60–79. Springer (2006)
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The sphincs+ signature framework. In: ACM CCS. pp. 2129–2146 (2019)
6. Beullens, W., Beyne, T., Udovenko, A., Vitto, G.: Cryptanalysis of the legendre prf and generalizations. IACR Transactions on Symmetric Cryptology pp. 313–330 (2020)
7. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: ASIACRYPT. pp. 464–492. Springer (2020)
8. Beullens, W., Delpech de Saint Guilhem, C.: Legroast: Efficient post-quantum signatures from the legendre prf. In: PQCrypto. pp. 130–150. Springer (2020)
9. Buser, M., Dowsley, R., Esgin, M., Gritti, C., Kasra Kermanshahi, S., Kuchta, V., Legrow, J., Liu, J., Phan, R., Sakzad, A., et al.: A survey on exotic signatures for post-quantum blockchain: Challenges and research directions. ACM Computing Surveys **55**(12), 1–32 (2023)
10. Buser, M., Liu, J.K., Steinfeld, R., Sakzad, A.: Post-quantum id-based ring signatures from symmetric-key primitives. In: ACNS. pp. 892–912. Springer (2022)
11. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: ASIACRYPT. pp. 234–252. Springer (2008)

12. Chator, A., Green, M., Tiwari, P.R.: Sok: Privacy-preserving signatures. Cryptology ePrint Archive (2023)
13. Ciampi, M., Persiano, G., Scafuro, A., Siniscalchi, L., Visconti, I.: Online/offline or composition of sigma protocols. In: EUROCRYPT. pp. 63–92. Springer (2016)
14. van Dam, W., Hallgren, S.: Efficient quantum algorithms for shifted quadratic character problems. arXiv preprint quant-ph/0011067 (2000)
15. Damgård, I.B.: On the randomness of legendre and jacobi sequences. In: ASIACRYPT. pp. 163–172. Springer (1988)
16. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: PQCrypto. pp. 419–440. Springer (2018)
17. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT. pp. 609–626. Springer (2004)
18. Esgin, M.F., Steinfeld, R., Zhao, R.K.: Matrict+: More efficient post-quantum private blockchain payments. In: IEEE S&P. pp. 1281–1298. IEEE (2022)
19. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: Matrict: efficient, scalable and post-quantum blockchain confidential transactions protocol. In: ACM CCS. pp. 567–584 (2019)
20. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: EUROCRYPT. pp. 186–194. Springer (1986)
21. Frixons, P., Schrottenloher, A.: Quantum security of the legendre prf. Mathematical Cryptology **1**(2), 52–69 (2021)
22. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Efficient set membership proofs using mpc-in-the-head. Cryptology ePrint Archive (2021)
23. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: A framework to compose-protocols for disjunctions. In: EUROCRYPT. pp. 458–487. Springer (2022)
24. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT. pp. 253–280. Springer (2015)
25. Guillou, L.C., Quisquater, J.J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: CRYPTO. pp. 216–231. Springer (1990)
26. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM JoC **39**(3), 1121–1152 (2009)
27. Kaludjerović, N., Kleinjung, T., Kostic, D.: Improved key recovery on the legendre prf. Cryptology ePrint Archive (2020)
28. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: ACM CCS. pp. 525–537 (2018)
29. Khovratovich, D.: Key recovery attacks on the legendre prfs within the birthday bound. Cryptology ePrint Archive (2019)
30. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: ICCSA. pp. 614–623. Springer (2005)
31. Lu, X., Au, M.H., Zhang, Z.: Raptor: a practical lattice-based (linkable) ring signature. In: ACNS. pp. 110–130. Springer (2019)
32. Lyubashevsky, V., Nguyen, N.K.: Bloom: Bimodal lattice one-out-of-many proofs and applications. In: ASIACRYPT. pp. 95–125. Springer (2022)
33. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Smile: set membership from ideal lattices with applications to ring signatures and confidential transactions. In: CRYPTO. pp. 611–640. Springer (2021)
34. May, A., Zweydinger, F.: Legendre prf (multiple) key attacks and the power of preprocessing. In: IEEE CSF. pp. 428–438. IEEE (2022)

35. Merkle, R.C.: A certified digital signature. In: ASIACRYPT. pp. 218–238. Springer (1989)
36. Naor, M.: Bit commitment using pseudo-randomness. In: ASIACRYPT. pp. 128–136. Springer (1989)
37. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT. pp. 552–565. Springer (2001)
38. Scafuro, A., Zhang, B.: One-time traceable ring signatures. In: ESORICS. pp. 481–500. Springer (2021)
39. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: PKC. pp. 166–180. Springer (2007)
40. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
41. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: ISPEC. pp. 48–60. Springer (2005)
42. Van Dam, W., Hallgren, S., Ip, L.: Quantum algorithms for some hidden shift problems. SIAM JoC **36**(3), 763–778 (2006)
43. 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. pp. 251–281. Springer (2021)

Appendix

A Merkle Tree based Accumulator

In this work, we use a Merkle hash proof system [35] as an accumulator scheme to accumulate vectors with high entropy. An accumulator scheme constructed from the Merkle tree consists of the following algorithms

1. $\mathtt{acc}_{\mathtt{pp}} \leftarrow \mathtt{Acc.Gen}(1^\lambda)$: On input a security parameter λ, the algorithm selects a hash function $\mathcal{H}_{\mathtt{MT}}$ from the collision-resistant hash function family, where $\mathcal{H}_{\mathtt{MT}} : \{0,1\}^{2\lambda} \times \{0,1\}^{2\lambda} \rightarrow \{0,1\}^{2\lambda}$ and returns $\mathtt{acc}_{\mathtt{pp}} := (\mathcal{H}_{\mathtt{MT}})$.
2. $\mathtt{acc} \leftarrow \mathtt{Acc.Eval}(\mathtt{acc}_{\mathtt{pp}}, Y)$: On input the public parameter $\mathtt{acc}_{\mathtt{pp}}$ and a set of ℓ (assume for simplicity that ℓ is a power of 2) elements $Y = \{y_1, \ldots, y_\ell\} \in \{0,1\}^{2\lambda \times \ell}$, compute the accumulated value $\mathtt{acc}$ as follows:
   - For each $j \in [\log(\ell)]$ and $s \in [\ell/2^j]$, compute $y_s^j = \mathcal{H}_{\mathtt{MT}}(y_{2s-1}^{j-1} \parallel y_{2s}^{j-1})$. Set $\mathtt{acc} = y_1^{\log(\ell)}$.
3. $\mathtt{auth} \leftarrow \mathtt{Acc.Proof}(\mathtt{acc}, Y, y_i)$: On input an accumulated value $\mathtt{acc}$, a set of ℓ elements $Y = (y_1, \ldots, y_\ell)$, and an element y_i, the algorithm computes the proof as follows:
   - Initialize $\mathtt{auth} = \{(i, \mathtt{sibling}(y_i^0))\}$, and for each $j \in [\log(\ell)]$, set $\mathtt{auth} = \mathtt{auth} \cup \{(\lceil i/2^j \rceil, \mathtt{sibling}(y_{\lceil i/2^j \rceil}^j))\}$. Return $\mathtt{auth}$ as the proof of $y_i \in Y$ being accumulated with accumulator $\mathtt{acc}$.
4. $0/1 \leftarrow \mathtt{Acc.Ver}(y_i, \mathtt{acc}, \mathtt{auth})$: On input an element y_i, an accumulator $\mathtt{acc}$, and an authentication path $\mathtt{auth}$, the algorithm parses $\mathtt{auth} = ((i_0, x^0), \ldots, (i_{\log(\ell)}, x^{\log(\ell)}))$, it computes the following:

- If i_0 is even, compute $y^1 = \mathcal{H}_{\mathtt{MT}}(y_i \parallel x^0)$, else compute $y^1 = \mathcal{H}_{\mathtt{MT}}(x^0 \parallel y_i)$.
- For each $j \in [\log(\ell)]$, if i_j is even, compute $y^j = \mathcal{H}_{\mathtt{MT}}(y^{(j-1)} \parallel x^j)$, else compute $y^j = \mathcal{H}_{\mathtt{MT}}(x^j \parallel y^{(j-1)})$
- If $y^{\log(\ell)} = \mathtt{acc}$, output 1. Else, output 0.

The Merkle tree based accumulator satisfies completeness (i.e., the verification of honestly computed proof $\mathtt{auth}$ will always return 1) and soundness (i.e., there is no polynomial time adversary can find an element $y' \notin Y$ such that $1 = \mathtt{Acc.Ver}(y', \mathtt{acc}, \mathtt{auth})$ with non-negligible probability, where $\mathtt{acc} = \mathtt{Acc.Eval}(\mathtt{acc}_{\mathtt{pp}}, Y)$).

B Formal Security Analysis of DualRing-PRF

In this subsection, we will present the unforgeability and anonymity proofs for DualRing-PRF. Unlike DualRing [43], which reduces the unforgeability to the special impersonation under key only attack, we directly reduce the unforgeability of DualRing-PRF to the hardness of β-approximate PRF relation. Namely, given a valid DualRing-PRF signature with respect to the challenge public key set $S^* = (pk_1^*, \ldots, pk_\ell^*)$, there exists an efficient extractor that extracts at least one of the witnesses. In the following security proof, we use $\mathfrak{B}(n, q)$ to denote the binomial distribution with n samples, each with success probability q. For two random variables A, B, we write $A \prec B$ if for all $x \in (-\infty, +\infty)$, we have $\Pr[A > x] \le \Pr[B > x]$.

Theorem 1. *Assuming the Merkle-tree based accumulator is binding and hiding, our ring signature scheme (DualRing-PRF) is unforgeable in the random oracle model, if the β-approximate PRF relation (Definition 3) is hard.*

Proof. Let hash functions defined in Algorithm 1 be modeled as random oracles and fix a constant $\beta \in [0, 1]$. Denote $\mathcal{A}$ as a PPT adversary breaking the unforgeability w.r.t. insider corruption of DualRing-PRF with probability ϵ. We build $\mathcal{B}$ that is given system parameters $\mathtt{pp}$, n challenge β-approximate PRF instances $S^* = (pk_1^*, \ldots, pk_n^*)$, outputs at least one β-approximate witness K_j^* corresponds to pk_j^* for $j \in [\ell]$.
Simulation of Random Oracles. $\mathcal{B}$ maintains a list of input-output pairs for random oracle queries and a list of "bad" outputs denoted as $\mathtt{Bad}_{\mathtt{H}}$ which stores all random oracle outputs. If the query was made before by $\mathcal{A}$, then $\mathcal{B}$ responses with the same answer. Otherwise, $\mathcal{B}$ returns a uniformly random output which is added to $\mathtt{Bad}_{\mathtt{H}}$, and records the new input-output pair in the list.
Simulation of Joining Oracle $\mathcal{JO}$. Assume $\mathcal{A}$ can query $\mathcal{JO}$ at most ℓ' times, where $\ell' \gg \ell$. $\mathcal{B}$ randomly choose a subset S^* contains ℓ indices. $\mathcal{B}$ assigns $pk_j^* \in S^*$ for $j \in [\ell]$ to these ℓ indices. For the remaining $\ell' - \ell$ indices, $\mathcal{B}$ generates the public key and private key pairs according to the algorithm. Upon j-th querying for $j \in [\ell']$, $\mathcal{B}$ returns the corresponding public key. Let $S = \{pk_1, \ldots, pk_{\ell'}\}$ be the set of registered public keys.

Simulation of Corruption Oracle $\mathcal{CO}$. On input pk, $\mathcal{B}$ checks $pk \in S^*$. If yes, $\mathcal{B}$ aborts.[9] Otherwise, $\mathcal{B}$ outputs the corresponding secret key.
Simulation of Signing Oracle $\mathcal{SO}$. On input a message M, a set of public keys $PK = (pk_1, \dots, pk_\ell)$, and the signer index π, $\mathcal{B}$ outputs $\perp$ if $pk_\pi \notin PK$. Furthermore, if $pk_\pi \in S/\{S^*\}$ then $\mathcal{B}$ returns $\sigma \leftarrow \texttt{Sign}(PK, M, sk_\pi)$.

Otherwise, let $\mathcal{H}_1$ be a programmable random oracle, $\mathcal{B}$ computes the following:

1. $\mathcal{B}$ randomly selects $(h_1, \dots, h_\ell) \xleftarrow{\$} \{0,1\}^{2\lambda}$ and
2. Expand the challenges $(I_{e,j}^{(1)}, \dots, I_{e,j}^{(B)}) \leftarrow \texttt{Expand}(h_j)$, where $I_{e,j}^{(b)} \in \mathcal{I}$ for $b \in [B]$, $j \in [\ell]$.
3. $\mathcal{B}$ randomly selects a "fake" secret key $K' \xleftarrow{\$} \mathbb{F}_p$ and follows the steps in Signing Phase 1 by replacing K with $K^{'}$ and computes $T_e^{(b)} = \mathcal{L}_0(o_e^{(b)}) - \sum_{j=1}^{\ell} pk_{I_{e,j}}^{(b)}$, where $o_e^{(b)} = (K' + I_{e,\pi}^{(b)})r_e^{(b)}$. Finally, $\mathcal{B}$ obtains σ_1.
4. $\mathcal{B}$ computes $c_1 = \bigoplus_{j\in[\ell]} h_j$ and programs $\mathcal{H}_1$ to set the output to c_1 with respect to the input $(\texttt{salt}, m, PK, \sigma_1)$. If $c_1 \in \texttt{Bad}_{\texttt{H}}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ continues to compute σ according to Phase 2-8 of the protocol.

Challenge. When $\mathcal{A}$ returns a forgery (PK', M', σ'), if $PK' \nsubseteq S^*$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ interprets

$$\begin{aligned}\sigma' = (&\texttt{salt}, c_2, c_4, (\texttt{perms}_e, \texttt{seeds}_e^{(\text{MPC})}, \texttt{seeds}_e^{(\text{MASK})}, \texttt{masks}_{e,\bar{k}_e},\\ &\texttt{auth}_{e,\bar{k}_e}, \Delta K_e, \Delta c_e, (o_e^{(b)}, \Delta I_{e,\bar{k}_e}^{(b)})_{b\in[B]}, \alpha_e, \beta_e, C_{e,\bar{i}_e}^{(\text{SD})},\\ &C_{e,\bar{k}_e,\bar{i}_e}^{(\text{MASK})})_{e\in[\tau]}, (h_j)_{j\in[\ell]}).\end{aligned}$$

$\mathcal{B}$ goes through the random oracle query transcript and finds the input-output query pair $((\texttt{salt}, e, i, \texttt{sd}_{e,i}^{(\text{MPC})}), C_{e,i}^{(\text{MPC})})$ for all $e \in [\tau], i \in [N]$. $\mathcal{B}$ computes:

$$([K_e]_i, [r_e^{(1)}]_i, \dots, [r_e^{(B)}]_i, [a_e]_i, [b_e]_i, [c_e]_i) \leftarrow \texttt{Expand}(sd_{e,i}^{(\text{MPC})})$$

Furthermore, $\mathcal{B}$ defines

$$\begin{aligned}(r_e^{(1)}, \dots, r_e^{(B)}, a_e, b_e) &\leftarrow \sum_{i=1}^{N}([r_e^{(1)}]_i, \dots, [r_e^{(B)}]_i, [a_e]_i, [b_e]_i)\\ K_e &\leftarrow \sum_{i=1}^{N}([K_e]_i) + \Delta K_e\\ c_e &\leftarrow \sum_{i=1}^{N}([c_e]_i) + \Delta c_e\end{aligned}$$

[9] We assume ℓ' is large enough such that the probability of $\mathcal{A}$ queries any public key in S^* is negligible. Specifically, let n_c be the number of queries $\mathcal{A}$ made to the corruption oracle, then $\ell' \gg n_c\ell$.

$\mathcal{B}$ checks if K_e is a β-approximate witness. If it is, $\mathcal{B}$ outputs K_e. Otherwise, $\mathcal{B}$ declares failure and exits. If $\mathcal{A}$ has running time T, then $\mathcal{B}$ runs in time $T + O(q_h + q_s)$, where q_h denotes the total number of queries to the random oracle and q_s denotes the total number of queries to the signing oracle.

We now show that if $\mathcal{A}$ wins the unforgeability w.r.t. insider corruption game with probability ε, the $\mathcal{B}$ outputs a β-approximate witness with probability at least $\varepsilon - e(q_h, q_s)$.

Cheating in Phase 1. Let $(Q_{\texttt{best}_1}, c_{\texttt{best}_1})$ be the best query-response pair that $\mathcal{A}$ received from $\mathcal{H}_1$. We define the meaning of "best" to be the query input-output pair that maximizes the number of $e \in [\tau]$ such that

$$\mathcal{L}_0^t((K_e + I_{e,\pi}^{(b)})r_e^{(b)}) = T_e^{(b)} + \sum_{j=1}^{n} pk_{I_{e,j}^{(b)}} \text{ for all } b \in [B]. \tag{4}$$

One can consider this situation as, for some (fixed and possibly invalid) witness K_e and randomness $r_e^{(b)}$ picked by $\mathcal{A}$, the residuosity (either quadratic or t-th power) of $(K_e + I_{e,\pi}^{(b)})r_e^{(b)}$, where $I_{e,\pi}^{(b)}$ is chosen uniformly at random, happens to be the same as $T_e^{(b)} + \sum_{j=1}^{n} pk_{I_{e,j}^{(b)}}$ for all $b \in [B]$. We show that if $\mathcal{B}$ outputs $\perp$, then the number of these "lucky" indices is bounded. More precisely, let $\#G_1(Q_1, c_1)$ be the number of parallel executions such that Equation (4) holds. Then we show $\#G_1(Q_{\texttt{best}_1}, c_{\texttt{best}_1})|_\perp \prec X^{(\mathcal{H}_1)}$, where $X^{(\mathcal{H}_1)}$ is defined as

$$X^{(\mathcal{H}_1)} = \max(X_1^{\mathcal{H}_1}, \ldots, X_{q_1}^{\mathcal{H}_1}) \tag{5}$$

for the number of random oracle queries to $\mathcal{H}_1$ being q_1, and $X_i^{(\mathcal{H}_1)}$ are i.i.d as $\mathfrak{B}(\tau, (1-\beta)^B)$.

For distinct queries to $\mathcal{H}_1$ of the form $Q_1 = (\texttt{salt}, m, \mathcal{PK}, \sigma_1)$, where

$$\sigma_1 \leftarrow (((C_{e,k,i}^{(\text{MASK})})_{k \in [M]}, C_{e,i}^{(\text{MPC})})_{i \in [N]}, (T_e^{(b)})_{b \in [B]}, \Delta K_e, \Delta c_e)_{e \in [\tau]}, \tag{6}$$

if Q_1 is defined in the random oracle query transcript, then let

$$\beta_e^{(b)}(Q_1) = d_H\left(\mathcal{L}_L^t(K_e) + (\mathcal{L}_0^t(r_e^{(b)}), \ldots, \mathcal{L}_0^{(t)}(r_e^{(b)})), T_e^{(b)} + \sum_{j=1}^{n} pk_{I_{e,b}^{(j)}}\right),$$

where d_H denotes the Hamming distance. Otherwise (Q_1 is not defined in the random oracle query transcript), $\beta_e^{(b)}(Q_1) = 1$. If $\mathcal{B}$ aborts, then none of K_e computed in the challenge phase is a β-approximate witness, meaning that $\beta_e^{(b)}(Q_1) > \beta$ for all $e \in [M]$, and $b \in [B]$. Since the challenge c_1 is chosen uniformly random, which implies that $h_\pi = c_1 \oplus \bigoplus_{j \in [n]/\{\pi\}} h_j$ is uniformly random. By the property of $\texttt{Expand}$ function, $(I_{e,\pi}^{(b)})_{b \in [B], e \in [\tau]}$ are also uniformly random from $\mathcal{I}$. Hence, the probability, for a certain e, such that Equation (4) holds is

$$\prod_{b \in [B]} (1 - \beta_e^{(b)}(Q_1)) \leq (1-\beta)^B.$$

Therefore, we have $\#G_1(Q_1, c_1)|_\perp \prec X_{Q_1}$, where $X_{Q_1} \sim \mathfrak{B}(\tau, (1-\beta)^B)$. Finally, since $G_1(Q_{\mathtt{best}_1}, c_{\mathtt{best}_1})$ is the maximum over at most q_1 queries to $\mathcal{H}_1$, it follows that $\#G_1(Q_{\mathtt{best}_1}, c_{\mathtt{best}_1})|\perp \prec X^{(\mathcal{H}_1)}$, where $X^{(\mathcal{H}_1)} = \max(X_1^{(\mathcal{H}_1)}, \dots, X_{q_1}^{(\mathcal{H}_1)})$, and $X_i^{(\mathcal{H}_1)} \sim \mathfrak{B}(\tau, (1-\beta)^B)$.
Cheating in Phase 2. Let $(Q_{\mathtt{best}_2}, c_{\mathtt{best}_2})$ be the best query-response pair that $\mathcal{A}$ received from $\mathcal{H}_2$. This is the pair where $\#G_2(Q_2, c_2)$ is maximum, where $Q_2 = (c_1, (\mathtt{acc}_e)_{e\in[\tau]})$. If there is no Q_1 such that (Q_1, c_1) was recorded in the query-response transcript of $\mathcal{H}_1$, then $G_2(Q_2, c_2)$ is bad (since this query cannot result in a valid siganture). Otherwise, let Q_1 be defined same as σ_1 (see Equation (6)). If there exists $(e, b) \in [M] \times [B]$ such that $\mathcal{L}_0^t(o_e^{(b)}) \neq T_e^{(b)} + \sum_{j=1}^{\ell} pk_{I_{e,j}^{(b)}}$, then this query cannot result in a valid signature either. Therefore, we define $G_2(Q_2, c_2)$ is the set of $e \in [\tau]/G_1(Q_1, c_1)$ for which

$$I_{e,\pi}^{(b)} \neq \mathtt{mask}_{e,\bar{k}_e} + \Delta I_{e,\bar{k}_e,\pi}^{(b)}. \tag{7}$$

Consider this being the case where $\mathcal{A}$ guesses the unrevealed mask correctly for some $e \in [\tau]$. Indeed, if $\mathcal{A}$ computes for more than 1 dishonest $\Delta I_{e,k,j}^{(b)}$ for any $j \in [\ell]$, it will get caught. Thus, it can manipulate the $\Delta I_{e,\bar{k}_e,\pi}^{(b)}$ for at most 1 execution to cause the proof generated in Phase 3 being accepted. Since $\bar{k}_e$ is chosen uniformly at random, the probability that the inequality (7) occurs is $1/M$. Note that $\mathcal{A}$ may also cheat in the execution of Merkle tree commitments by finding collisions. However, compare to $1/M$, the probability of finding collisions is negligible. Thus, if $\mathcal{B}$ outputs $\perp$, the number of good indices is bounded: $\#G_2(Q_2, c_2)|_{\#G_1(Q_1,c_1)=\tau_1'} \prec \tau_1 + X_{Q_2}$ where $X_{Q_2} \sim \mathfrak{B}(\tau - \tau_1', 1/M)$. Since for integers $a \leq b$ and $q \in [0, 1]$, we have $\mathfrak{B}(b, q) \prec a + \mathfrak{B}(b - a, q)$. This implies that $\#G_2(Q_2, c_2)|_{\#G_1(Q_{\mathtt{best}_1}, c_{\mathtt{best}_1})=\tau_1} \prec \tau_1 + X_{Q_2}$, where $X_{Q_2} \sim \mathfrak{B}(\tau - \tau_1, 1/M)$. Since $\#G_2(Q_{\mathtt{best}_2, c_{\mathtt{best}_2}})$ is the maximum over at most q_2 values of $G_2(Q_2, c_2)$, it follows that

$$\#G_2(Q_{\mathtt{best}_2} c_{\mathtt{best}_2})|_{\#G_1(Q_{\mathtt{best}_1}, c_{\mathtt{best}_1})=\tau_1} \prec \tau_1 + X^{(\mathcal{H}_2)}.$$

Finally, by conditioning on $\perp$ and summing over all τ_1, we get

$$\#G_2(Q_{\mathtt{best}_2} c_{\mathtt{best}_2})|_\perp \prec \#G_1(Q_{\mathtt{best}_1} c_{\mathtt{best}_1})|_\perp + X^{(\mathcal{H}_2)} \prec X^{(\mathcal{H}_1)} + X^{(\mathcal{H}_2)},$$

where $X^{(\mathcal{H}_2)} = \max(X_1^{(\mathcal{H}_2)}, \dots, X_{q_2}^{(\mathcal{H}_2)})$ and $X_i^{(\mathcal{H}_2)} \sim \mathfrak{B}(\tau - X^{(\mathcal{H}_1)}, 1/M)$.
Cheating in Phase 3. Let $\#G_3(Q_{\mathtt{best}_3}, c_{\mathtt{best}_3})$ be the best query-response pair $\mathcal{A}$ received from $\mathcal{H}_3$, which is the pair for which $\#G_3(Q_3, c_3)$ is maximum, for $Q_3 = (c_2, (o_e^{(b)})_{b\in[B], e\in[\tau]}$. If there does not exist Q_2 such that (Q_2, c_2) is queried in $\mathcal{H}_2$, then all indices are bad. Otherwise, let $Q_2 = (c_1, (\mathtt{acc}_e)_{e\in[\tau]})$. If there exists accumulated $\Delta I_{e,k,j}^{(b)}$ for $(j, e, k, b) \in [\ell] \times [\tau] \times [M] \times [B]$ and $k \neq \bar{k}_e$, such that $I_{e,j}^{(b)} \neq \mathtt{mask}_{e,k} + \Delta I_{e,k,j}^{(b)}$, then we say $G_3(Q_3, c_3) = \{\}$, since this results in an invalid siganture. Otherwise, we say $G_3(Q_3, c_3)$ is the set of executions $e \in [\tau]$ for which $\alpha_e \cdot \beta_e = \gamma_e$. We prove that in the case that $\mathcal{B}$ outputs $\perp$, the number

of good indices is bounded. Note that for fixed a_e, b_e, c_e, K_e, $(r_e^{(b)})_{b\in[B]}, \mathtt{mask}_e$ and $(\Delta I_{e,k,j}^{(b)})_{k\in[M],j\in[\ell],b\in[B]}$, the function

$$\alpha_e(\epsilon_e)\beta_e(\lambda_e^{(b)}) - \gamma_e(\epsilon_e, \lambda_e^{(b)})$$

is a quadratic polynomial in $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)})$. Moreover, this is the zero polynomial iff $c_e = a_e b_e$ and $o_e^{(b)} = (K_e + (\mathtt{mask}_{e,\bar{k}_e} + \Delta I_{e,\bar{k}_e,\pi}^{(b)}))r_e^{(b)}$ for all $b \in [B]$. If $e \notin G_2(Q_2, c_2)$ and $e \notin G_1(Q_1, c_1)$, then to make sure $\mathcal{L}_0(o_e^{(b)}) = T_e^{(b)} + \sum_{j=1}^{\ell} pk_{I_{e,j}^{(b)}}$, $\mathcal{A}$ computes $o_e^{(b)} \neq (K_e + (\mathtt{mask}_{e,\bar{k}_e} + \Delta I_{e,\bar{k}_e,\pi}^{(b)}))r_e^{(b)}$, where $\mathtt{mask}_{e,\bar{k}_e} + \Delta I_{e,\bar{k}_e,\pi}^{(b)} = I_{e,\pi}^{(b)}$. By Schwartz-Zipple lemma, where a uniformly random choice of $c_3 = \{\epsilon_e, \lambda_e^{(b)}\}_{e\in[\tau],b\in[B]} \in \mathbb{F}_p^{\tau(1+B)}$, the probability that $e \in G_3(Q_3, c_3)$ is at most $2/p$. From the similar arguments in Phase 2, we have that

$$\begin{aligned}\#G_3(Q_{\mathtt{best}_3} c_{\mathtt{best}_3})|_\perp &\prec \#G_2(Q_{\mathtt{best}_2} c_{\mathtt{best}_2})|_\perp + X^{(\mathcal{H}_3)} \\ &\prec X^{(\mathcal{H}_1)} + X^{(\mathcal{H}_2)} + X^{(\mathcal{H}_3)},\end{aligned}$$

where $X^{(\mathcal{H}_3)} = \max(X_1^{(\mathcal{H}_3)}, \ldots, X_{q_3}^{(\mathcal{H}_3)})$ and $X_i^{(\mathcal{H}_3)} \sim \mathfrak{B}(\tau - X^{(\mathcal{H}_1)} - X^{(\mathcal{H}_2)}, 2/p)$.
Cheating in Phase 4. Define $G_4(Q_4, c_4)$ being the set of $e \in [M]$ that are good, where $Q_4 = (c_3, (\alpha_e, \beta_e, ([\alpha_e]_i, [\beta_e]_i, [\gamma_e]_i)_{i\in[N]})_{e\in[\tau]})$ that $\mathcal{A}$ makes to $\mathcal{H}_4$. If there does not exist (Q_3, c_3) that was queried to $\mathcal{H}_3$, then all indices are bad as this leads to invalid signature. Consider e that was not in any of $G_i(Q_i, c_i)$ for $i \in [1, 3]$, then to make the signature accepted, $\mathcal{A}$ must guess $\bar{i}_e$ (the unopened party) correctly, with probability $1/N$ as $\bar{i}_e$ is chosen uniformly from $[N]$. In fact, if there are less than $N-1$ honest parties, then this will be caught by the verifier, which results in an invalid signature. If there are N honest parties, then $\gamma_e \neq \alpha_e\beta_e$, which also results in invalid signature. Following from previous arguments, we have that

$$\begin{aligned}\#G_4(Q_{\mathtt{best}_4} c_{\mathtt{best}_4})|_\perp &\prec \#G_3(Q_{\mathtt{best}_3} c_{\mathtt{best}_3})|_\perp + X^{(\mathcal{H}_4)} \\ &\prec X^{(\mathcal{H}_1)} + X^{(\mathcal{H}_2)} + X^{(\mathcal{H}_3)} + X^{(\mathcal{H}_4)},\end{aligned}$$

where $X^{(\mathcal{H}_4)} = \max(X_1^{(\mathcal{H}_4)}, \ldots, X_{q_4}^{(\mathcal{H}_4)})$ and $X_i^{(\mathcal{H}_4)} \sim \mathfrak{B}(\tau - X^{(\mathcal{H}_1)} - X^{(\mathcal{H}_2)} - X^{(\mathcal{H}_3)}, 1/N)$.
Abort Probability We now analyze the abort probability of $\mathcal{B}$ due to collision. In other words, $\mathcal{B}$ aborts if there exists a randomly chosen output that was already recorded in $\mathtt{Bad_H}$. To do this, we first analyze the number of entries that can be recorded in $\mathtt{Bad_H}$. The random oracles simulated by $\mathcal{B}$ are $\mathcal{H}_{\mathtt{sd}}, \mathcal{H}_{\mathtt{mask}}, \mathcal{H}_{\Delta}, \mathcal{H}_{\mathtt{MT}}, \mathcal{H}_{\mathtt{acc}}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ and $\mathcal{H}_4$.

- **Phase 1.** Let $q_{\mathtt{sd}}$ and $q_{\mathtt{mask}}$ denote the number of queries $\mathcal{A}$ made to $\mathcal{H}_{\mathtt{sd}}$ and $\mathcal{H}_{\mathtt{mask}}$ in Phase 1, respectively. On every input query to $\mathcal{H}_{\mathtt{sd}}$ ($\mathcal{H}_{\mathtt{mask}}$), $\mathcal{B}$ randomly selects a input $x \in \{0,1\}^{2\lambda}$ and adds x to $\mathtt{Bad_H}$. Hence, Phase 1 adds $q_{\mathtt{sd}} + q_{\mathtt{mask}}$ entries to $\mathtt{Bad_H}$.

- **Phase 2.** Let q_1 be the number of queries $\mathcal{A}$ made to $\mathcal{H}_1$, where the input query is of the form $(\texttt{salt}, m, \mathcal{PK}, \sigma_1)$. $\mathcal{B}$ interprets $\sigma_1 = (((C^{(\text{MASK})}_{e,k,i})_{k\in[M]}, C^{(\text{MPC})}_{e,i})_{i\in[N]}, \dots)_{e\in[\tau]}$ and adds $C^{(\text{MASK})}_{e,k,i}$, $C^{(\text{MPC})}_{e,i}$ to $\texttt{Bad}_\texttt{H}$. $\mathcal{B}$ randomly selects $x \in \{0,1\}^{2\lambda}$ as the response and adds x to $\texttt{Bad}_\texttt{H}$. Let q_Δ and $q_{\texttt{acc}}$ be the number of queries to $\mathcal{H}_\Delta$ and $\mathcal{H}_{\texttt{acc}}$, respectively. Let $q_{\texttt{MT}}$ be the number of queries to the random oracle that generates the Merkle tree. Then, the total number of entries that are added to $\texttt{Bad}_\texttt{H}$ is $(\tau \cdot N \cdot (M+1)+1)q_1 + q_\Delta + 3q_{\texttt{MT}} + (M+1)q_{\texttt{acc}}$.
- **Phase 3.** Let q_2 be the number of queries to $\mathcal{H}_2$. Then for the query of the form $(c_1, (\texttt{acc}_e)_{e\in[\tau]})$, $\mathcal{B}$ samples $x \in \{0,1\}^{2\lambda}$ and adds at most $(\tau+2)q_2$ entries to $\texttt{Bad}_\texttt{H}$.
- **Phase 5.** Let q_3 be the number of queries to $\mathcal{H}_3$. Then for the query of the form $(c_2, \dots)$, $\mathcal{B}$ samples $x \in \{0,1\}^{2\lambda}$ and adds at most $2q_3$ entries to $\texttt{Bad}_\texttt{H}$.
- **Phase 7.** Let q_4 be the number of queries to $\mathcal{H}_4$. Then for the query of the form $(c_3, \dots)$, $\mathcal{B}$ samples $x \in \{0,1\}^{2\lambda}$ and adds at most $2q_4$ entries to $\texttt{Bad}_\texttt{H}$.

For the aborting probability of the signing oracle, the case that $\mathcal{B}$ aborts is the simulated σ_1 was queried before. Since $\texttt{salt}$ is selected uniformly random for every signature generation, the probability that σ_1 was queried before is at most $\frac{q_1 q_s}{2^{2\lambda}}$, where q_s denotes the number of signing oracle queries. Let the (maximum) size of $\texttt{Bad}_\texttt{H}$ be denoted as $q_h = q_{\texttt{sd}} + q_{\texttt{mask}} + q_\Delta + 3q_{\texttt{MT}} + (M+1)q_{\texttt{acc}} + \tau \cdot N \cdot (M+1)+1)q_1 + (\tau+2)q_2 + 2q_3 + 2q_4$, the aborting probability of $\mathcal{B}$ owing to collision is then $\frac{q_h^2 + q_1 q_s}{2^{2\lambda}}$.
To Conclude. If $\mathcal{A}$ wins the unforgeability game w.r.t. insider corruption with probability ϵ and challenge set S^*, then $\mathcal{B}$ outputs a β-approximate witness with probability $\epsilon - e(q_h, q_s)$, where

$$e(q_h, q_s) = \frac{q_h^2 + q_1 q_s}{2^{2\lambda}} + \Pr[X^{(\mathcal{H}_1)} + X^{(\mathcal{H}_2)} + X^{(\mathcal{H}_3)} + X^{(\mathcal{H}_4)} = \tau]$$

Indeed, by the law of total probability we have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ aborts}] \\
&\quad + \Pr[\mathcal{A} \text{ wins}|_\perp] + \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs witness}] \\
&\leq \Pr[\mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins}|_\perp] + \Pr[\mathcal{B} \text{ outputs witness}] \\
&\leq e(q_h, q_s) + \Pr[\mathcal{B} \text{ outputs witness}]
\end{aligned}$$

Theorem 2. *Our ring signature scheme (DualRing-PRF) is anonymous against full key exposure in the random oracle model.*

Proof. Let hash functions defined in Algorithm 1 be modeled as random oracles. $\mathcal{B}$ runs signature setup to generate $\texttt{pp}$ and gives $\texttt{pp}$ to $\mathcal{A}$. $\mathcal{B}$ simulates the oracles similarly as in unforgeability game. Except that in anonymity game, the key generation oracle generates all keys using the algorithm $\texttt{KGen}$ (i.e., there is no challenge key sets). Furthermore, $\mathcal{A}$ is allowed to corrupt all keys generated from $\mathcal{KO}$.
Challenge. $\mathcal{A}$ gives $\mathcal{B}$ a message m^*, a set of public keys $\mathcal{PK}^*$, and two signer indices i_0, i_1 w.r.t. the same ring $\mathcal{PK}^*$. $\mathcal{B}$ simulates the signature as follows:

1. $\mathcal{B}$ picks random challenges for each ring member $h_1, \dots, h_\ell \in \{0,1\}^\lambda$, where $\ell = |\mathcal{PK}^*|$.
2. $\mathcal{B}$ expands the challenges to obtain $(I^{(1)}_{e,j}, \dots, I^{(B)}_{e,j})_{e\in[\tau]}$ for $j \in [\ell]$.
3. $\mathcal{B}$ randomly selects responses $o^{(b)}_e \xleftarrow{\$} \mathbb{F}_p$ and computes $T^{(b)}_e = \mathcal{L}^t_0(o^{(b)}_e) - \sum_{j=1}^{\ell} pk_{I^{(b)}_{e,j}}$ for all $b \in [B]$and $e \in [\tau]$. (i.e., this step makes sure that the residuosity check passes).
4. $\mathcal{B}$ randomly samples a "fake" secret key K' and computes σ_1 following the description of Phase 1, except that $(T^{(b)}_e)_{b\in[B],e\in[\tau]}$ is computed as in previous step.
5. $\mathcal{B}$ computes $c_1 = h_1 \oplus \dots \oplus h_\ell$ and programs the oracle $\mathcal{H}_1$ to output c_1 on input $(\texttt{salt}, m^*, \mathcal{PK}^*, \sigma_1)$. If $(\texttt{salt}, m^*, \mathcal{PK}^*, \sigma_1)$ was queried before, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ continues.
6. $\mathcal{B}$ randomly samples c_2 and $(\bar{k}_e)_{e\in[\tau]} \leftarrow \texttt{Expand}(c_2)$.
7. For every $e \in [\tau]$ and $k \in [M]/\{\bar{k}_e\}$, $\mathcal{B}$ follows the protocol honestly.
8. For $k = \bar{k}_e$, $\mathcal{B}$ does the following:
   (a) Randomly selects $(I^{(1)}_{e,\bar{k}_e}, \dots, I^{(B)}_{e,\bar{k}_e}) \xleftarrow{\$} \mathcal{I}$.
   (b) Computes $\Delta I^{(b)}_{e,\bar{k}_e}$ such that $o^{(b)}_e = (K' + \texttt{mask}_{e,\bar{k}_e} + \Delta I^{(b)}_{e,\bar{k}_e}) r^{(b)}_e$. Note that $o^{(b)}_e$ and K' was the fake response and witness chosen in the first phase.
   (c) Commits to $\Delta I^{(b)}_{e,\bar{k}_e}$ according to the protocol and accumulates the commitment with $\ell - 1$ garbage values to obtain $\texttt{acc}_{e,\bar{k}_e}$.
9. Follow the protocol to obtain $\texttt{acc}_e$ for all $e \in [\tau]$.
10. Program the random oracle $\mathcal{H}_2$ such that on input $(c_1, (\texttt{acc}_e)_{e\in[\tau]})$ it outputs c_2 that was chosen previously. If $c_2 \in \texttt{Bad}_\texttt{H}$. If $(c_1, (\texttt{acc}_e)_{e\in[\tau]})$ was queried before, then $\mathcal{B}$ aborts.
11. Follow the rest of the protocol to obtain the signature σ.

$\mathcal{B}$ outputs σ to $\mathcal{A}$ and a randomly chosen bit *bit*
Output $\mathcal{A}$ outputs a guess *bit*$'$. Observe that *bit* is not used in the generation of σ, hence $\mathcal{A}$ wins with probability 1/2.
Probability Analysis. We now analyze the probability that $\mathcal{B}$ does not abort in the above simulation. For q_1 queries made to $\mathcal{H}_1$, q_2 queries made to $\mathcal{H}_2$, and q_s queries made to signing oracle, $\mathcal{B}$ aborts with the following probability. Firstly, due to the uniformly random salt of length 2λ bit, with probability $\leq q_1/2^{2\lambda} \leq (q_1 + q_s)/2^{2\lambda}$ for each signing query. Hence, over all q_s signing queries with probability $\leq q_s \cdot (q_1 + q_s)/2^{2\lambda}$. Similarly if c_1 is uniform in $2^{2\lambda}$ for each sign query, then the bad event happens with probability $\leq q_s \cdot (q_2 + q_s)/2^{2\lambda}$. Then the total abort probability is $\leq q_s \cdot (q_1 + q_2 + 2q_s)/2^{2\lambda}$.

Hence, if $\mathcal{B}$ does not abort, then no PPT adversary $\mathcal{A}$ can win the anonymity game with probability over 1/2.

C Formal Security Analysis of DualRing$_L$-PRF

Theorem 3. *(Linkability) Our linkable ring signature (DualRing$_L$-PRF) is linkable in the random oracle model, if the β-approximate PRF relation (Definition 3) is hard.*

Proof. We try to derive some contradictions based on the assumption that $\mathcal{A}$ can produce two valid signatures that are unlinked with just one private key. We use the same setting as the proof in Theorem 1, except that we only allow the adversary $\mathcal{A}$ query the corruption oracle $\mathcal{CO}$ once. Namely, $\mathcal{A}$ only knows one secret key. Let us assume that the corrupted secret key has index $\pi \in [\ell]$. In the proof, we show that if $\mathcal{A}$ is able to computes two signatures with respect to the same event that are unlinked (under the condition where $\mathcal{A}$ only knows one secret key), then there exists an extractor algorithm $\mathcal{B}$ that extracts two valid β-approximate witnesses of the Legendre/power residue PRFs.

Suppose $\mathcal{A}$ produces two valid signatures σ_1, σ_2 with respect to the rings $\mathcal{PK}_1, \mathcal{PK}_2$ and messages m_1, m_2 which are valid and unlinked (i.e., $\texttt{tag}_1 \neq \texttt{tag}_2$). The extractor $\mathcal{B}$ follows the steps described in the proof of Theorem 1 and extracts two valid secret keys K_1 and K_2.

Since $\texttt{tag}_1 \neq \texttt{tag}_2$, then with overwhelming probability that $K_1 \neq K_2$, which contradicts with the assumption that $\mathcal{A}$ only knows one secret key.

Theorem 4. *(Non-Slanderability) Our linkable ring signature scheme is nonslanderable in the random oracle mode, if the β-approximate PRF relation (Definition 3) is hard.*

Proof. Suppose the adversary $\mathcal{A}$ can query any oracle except that it cannot submit a chosen public key pk_π to $\mathcal{CO}$. $\mathcal{A}$ gives $\mathcal{B}$ pk_π, a list $\mathcal{PK}$ such that $pk_\pi \in \mathcal{PK}$,

Let us assume that $\mathcal{B}$ generates a linkable ring signature σ using sk_π with respect to $\mathcal{PK}$, a message m, and an event ID e_{ID}. In return, $\mathcal{B}$ generates a signature $\sigma = (\ldots, \texttt{tag})$ using sk_π and gives it to $\mathcal{A}$. $\mathcal{A}$ continues to query oracles except it is not allowed to submit pk_π to $\mathcal{CO}$. Suppose $\mathcal{A}$ produces a valid signature σ' that is not the output from $\mathcal{SO}$ and is linked with σ. Then, from Theorem 1 and Theorem 3, the extractor can extract a valid witness sk_π corresponding to pk_π, which contradicts with the assumption.

D Signature Size Analysis

We begin by demonstrating the estimation of signature size in DualRing-PRF. A signature is composed of the following elements:

- 2λ-bits salt $\texttt{salt}$ and challenges h_2, h_4.
- λ-bits $2 \times \log(M)$ seeds $\texttt{perms}_e$ and $\texttt{seeds}_e^{(\text{MASK})}$, where $e \in [\tau]$.
- λ-bits $2 \times \log(N)$ seeds $\texttt{masks}_{e,\bar{k}_e}$ and $\texttt{seeds}_e^{(\text{MPC})}$, where $e \in [\tau]$.
- Field elements $\Delta K_e, \Delta z_e, \Delta c_e, (o_e^{(b)}, \Delta I_{e,\bar{k}_e})_{b\in[B]}, \alpha_e$ and β_e. According to our choice of parameter, the field element is of size approximately λ-bits.
- 2λ-bits commitments $C_{e,\bar{i}_e}^{(\text{SD})}$ and $C_{e,\bar{k}_e,\bar{i}_e}^{(\text{MASK})}$ for $e \in [M]$.
- ℓ λ-bis challenges, where ℓ is the size of the ring.

Thus, the signature size $|\sigma|$ of DualRing-PRF is:

$$|\sigma| = 6\lambda + \tau \cdot \lambda(2\log M + 2\log N + 2B + 9) + \ell\lambda$$

Our Linkable ring signature adds additional $B \cdot \tau$ responses to the proof, denoted as $(o_e^{(b)\prime})_{b \in [B], e \in [\tau]}$, where each response is approximately λ-bits, and the linking tag is of the same size as the public key. Hence, the size $|\sigma'|$ for DualRing_L-PRF is:

$$|\sigma'| = |\sigma| + B\tau\lambda + L \cdot \log(t)$$