Efficient Constant-Size Linkable Ring Signatures for Ad-Hoc Rings via Pairing-Based Set Membership Arguments

Min Xie
Harbin Institute of Technology, Shenzhen
Shenzhen, China
minxie@stu.hit.edu.cn

Zhengzhou Tu
Harbin Institute of Technology, Shenzhen
Shenzhen, China
23S051021@stu.hit.edu.cn

Man Ho Au
The Hong Kong Polytechnic University
Hong Kong, China
man-ho-allen.au@polyu.edu.hk

Junbin Fang
Jinan University
Guangzhou, China
tjunbinfang@jnu.edu.cn

Xuan Wang
Harbin Institute of Technology, Shenzhen
Shenzhen, China
Guangdong Key Laboratory of New Security and Intelligence Technology
Shenzhen, China
wangxuan@cs.hitsz.edu.cn

Zoe Lin Jiang*
Harbin Institute of Technology, Shenzhen
Shenzhen, China
Guangdong Key Laboratory of New Security and Intelligence Technology
Shenzhen, China
zoeljiang@hit.edu.cn

Abstract

Linkable Ring Signatures (LRS) allow users to anonymously sign messages on behalf of ad-hoc rings, while ensuring that multiple signatures from the same user can be linked. This feature makes LRS widely used in privacy-preserving applications like e-voting and e-cash. To scale to systems with large user groups, efficient schemes with short signatures and fast verification are essential. Recent works, such as DualDory (ESORICS'22) and LLRing (ESORICS'24), improve verification efficiency through offline precomputations but rely on static rings, limiting their applicability in ad-hoc ring scenarios. Similarly, constant-size ring signature schemes based on accumulators face the same limitation.

In this paper, we propose a framework for constructing constant-size LRS suitable for large ad-hoc rings. We introduce a novel pairing-based Set Membership Argument (SMA) with a proof size of only three group elements. By leveraging KZG polynomial commitments, we optimize the verification to require only constant group exponentiations and pairings, as well as linear field multiplications. Utilizing the SMA, our framework achieves constant-size signatures with verification dominated by linear field operations, outperforming existing schemes that require linear group exponentiations in ad-hoc ring settings. Moreover, it exhibits strong scalability: (i) compatibility with any PKI-based cryptosystem and (ii) scoped linkability, enabling flexible definitions of linking scope.

We instantiate our framework using a discrete logarithm public key structure. On the *BN*254 curve, our signature size is fixed at 687 bytes, which to our best knowledge is the shortest LRS for ring sizes larger than 32. For a ring size of 1024, our verification cost is only 10.4 ms, achieving 48.6×, 2.6×–467×, 7.9×–13.2×, and 2.2×–102.5× improvements over Omniring (CCS'19), DualDory (with and without precomputation), LLRing-DL (with and without precomputation), and LLRing-P (with and without precomputation), respectively. Moreover, this performance gap continues to grow as the ring size increases.

CCS Concepts

• **Security and privacy → Digital signatures**.

Keywords

Linkable Ring Signatures, Ad-hoc Rings, Set Membership Arguments, Blockchain

ACM Reference Format:
Min Xie, Zhengzhou Tu, Man Ho Au, Junbin Fang, Xuan Wang, and Zoe Lin Jiang. 2025. Efficient Constant-Size Linkable Ring Signatures for Ad-Hoc Rings via Pairing-Based Set Membership Arguments. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei.* ACM, New York, NY, USA, 23 pages. https://doi.org/10.1145/3719027.3744830

1 Introduction

Ring signatures, first introduced by Rivest et al. [39], allow a user to anonymously sign a message on behalf of an ad-hoc ring, without revealing the identity of the actual signer. This feature makes ring signatures a crucial privacy protection tool, with applications in scenarios such as anonymous whistleblowing and privacy-preserving transactions. Compared to group signatures [11], ring signatures do not rely on a centralized manager, allowing members to form ad-hoc rings spontaneously and achieve more flexible anonymity.

To strengthen ring signatures, linkable ring signatures (LRS) incorporate linkability, allowing the detection of multiple signatures from the same signer. This property prevents issues such as

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '25, Taipei.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744830

duplicate votes in electronic voting and double-spending in privacy-preserving transactions. Currently, linkable ring signatures have been widely adopted in blockchain platforms to enable confidential transactions. For instance, Zerocash [5] and Monero [37] utilize linkable ring signatures to unlink the sender and receiver in a transaction while employing a tag that is unique to each signer to prevent double-spending with the same key.

However, as the scale of blockchain applications increases, the size of ring signatures and the computational overhead for verification pose significant challenges to blockchain storage and performance. Specifically, the increase in signature size directly leads to a dramatic increase in blockchain storage requirements, affecting node synchronization efficiency. In addition, the computational complexity of verifying a ring signature typically grows linearly with the ring size. In large ring scenarios, this verification overhead can severely degrade transaction processing efficiency and impose high costs on miners. These challenges motivate the exploration of designing (linkable) ring signatures with smaller signature sizes and more efficient verification.

1.1 Related Works

Zero-Knowledge Proof Based Approach. This approach is considered one of the most mainstream methods for constructing ring signatures, as it allows proving knowledge of the secret key corresponding to the signer index within the ring. One prominent scheme, One-out-of-Many [21], achieves logarithmic proof size, although its proving time is super-linear. Similar Bulletproofs-based schemes, such as Omniring [29] and DualRing [41], also achieve logarithmic proof sizes. However, they still require $\mathcal{O}(n)$ group exponentiations for verification (n denotes the ring size). To improve verification efficiency, recent works such as DualDory [6] and LLRing [24] have proposed leveraging offline precomputations by the verifier. However, this approach is only feasible if the static ring is known to the verifier in advance, before the signer generates the ring signature. These ring signature schemes maintain a logarithmic size and generally require designing specialized zero-knowledge proofs (ZKP) for specific cryptographic systems, such as those based on discrete logarithms (DL) or RSA.

SNARK-Based Approach. It can be considered a special case of the ZKP based approach by utilising the powerful succinct non-interactive arguments of knowledge (SNARKs) [12, 20]. Concretely, the signer first signs the message using any signature scheme, then directly uses a SNARK to prove that (1) the correctness of the signature generation; and (2) the public key is included in the ring. (2) can be further optimised by arranging the ring of public keys into a Merkle tree (MTree) and uses a SNARK to prove that the corresponding public key is a leaf of the tree. While in principle it leads to constant-size ring signature (if the underlying SNARK is constant-size), the verification cost is very high. The main reason is that the verifier is required to check that the Merkle root is computed correctly by recomputing the root from the ring of public keys. The cost can easily be 1 to 2 orders of magnitude worse than existing approaches. For concreteness, we give a more elaborated analysis on the SNARK + MTree approach in Appendix C.

Accumulator-Based Approach. Constant-size ring signatures can be constructed using accumulators [7, 14, 36]. An accumulator allows the signer to "compress" n public keys into a constant-size value, along with a witness that proves the signer's public key is part of the public key set. By utilizing zero-knowledge proofs, the accumulator can demonstrate that it opens to a specific public key and provide proof of knowledge of the corresponding private key. However, the use of accumulators introduces two key limitations. First, it relies on a trusted setup. Second, it is typically only applicable to static rings, where the accumulator value can be precomputed; otherwise, the verifier must recompute the accumulator value using $\mathcal{O}(n)$ group exponentiations or hash operations. In addition, RSA-based schemes [14] require a prime-based public key structure, and some DL-based schemes [7] rely on composite-order groups, resulting in larger signature sizes and inefficient verification.

1.2 Our Result

In this paper, we present a new framework for constructing constant-size linkable ring signatures with efficient verification for large ad-hoc rings. Unlike the constant-size ring signature framework based on accumulators, we follow a ZKP-based approach, which separates the index of the signer in the ring from the proof that demonstrates knowledge of the corresponding private key. To handle the index relationship for the ring, we design a novel pairing-based set membership argument (SMA) that requires only three group elements in its proof. In particular, the unique design of the SMA's public parameters and the form of the verifier's expensive computations (group exponentiations) allow these expensive operations to be outsourced to the signer. This is achieved by leveraging KZG polynomial commitments [26], enabling the verifier to perform only a constant number of expensive operations, such as group exponentiations and pairings, as well as linear field multiplications in $\mathbb{Z}_p$. Importantly, unlike DualDory [6] and LLRing [24], our verification optimization supports ad-hoc rings rather than being limited to static rings, meaning the verifier does not need to know the ring in advance.

Moreover, we capture the fact that the statements corresponding to private key relations are ring independent, allowing us to leverage the benefits of SNARKs—such as constant-time verification and constant-size proofs—by designing a fixed circuit. This ring-independent design eliminates the need for frequent trusted setups in SNARKs. In addition, by using cryptographic hash functions to map public keys to a single domain, our proposed SMA becomes compatible with any Public Key Infrastructure (PKI)-based cryptosystem, as the proof of the public-private key pair relationship is proven by SNARKs.

Finally, we introduce linkability to our LRS through deterministic tags. This feature is incorporated by introducing a proof of correctness for tag computation within the SNARKs framework. We explore the variant known as scoped linkability (or event-oriented linkability, as described in [40]). A scoped LRS enables the linking of signatures generated with the same key and scoped string. Notably, scoped linkability is a more general form of linkability: by setting the scoped string to the ring or the message, the LRS supports ring-based [33] or message-based linkability [17].

Contributions. Our contributions are summarized as follows.

- We propose a constant-size SMA with a proof size of only 3 group elements. By leveraging KZG polynomial commit-

ments, we optimize the verification process of SMA, requiring only constant expensive operations (group exponentiations and pairing) and $\mathcal{O}(n)$ field multiplications in $\mathbb{Z}_p$, with an additional overhead of 4 group elements in the proof.
- Using our SMA, we introduce a new framework for constructing constant-size linkable ring signatures with constant expensive operations and $\mathcal{O}(n)$ field multiplications for the verifier. In the ad-hoc ring setting, our verification complexity achieves the best performance(see Table 1), whereas other schemes require at least $\mathcal{O}(n)$ group exponentiations[1]. Moreover, our framework exhibits strong scalability: (i) it is compatible with any existing PKI-based cryptosystem; and (ii) it supports scoped linkability, allowing flexibility in defining the scope of linkability.
- We provide rigorous security proofs for our SMA and framework under the Random Oracle Model (ROM) and the Algebraic Group Model (AGM).
- We instantiate our linkable ring signature using a DL structure and conducte evaluations on the *BN*254 curve. The signature size remains 687 bytes, regardless of the ring size. Among representative ring signature schemes (see Table 2), our signature is the shortest when $n \geq 512$. For linkable ring signature schemes, our signature is the shortest when $n \geq 32$. When $n = 1024$, our verification cost is only 10.4 ms. In comparison to state-of-the-art (SOTA) ring signature schemes (see Fig. 1), Omniring [29] requires 505.11 ms for verification, DualDory [6] takes 26.67 ms with precomputation (4857.01 ms without), LLRing-DL [24] takes 82.44 ms with precomputation (137.63 ms without), and LLRing-P [24] takes 23.25 ms with precomputation (1065.7 ms without). For $n = 65536$, our scheme achieves a signing time of 3.27 s and a verification time of 26.29 ms, making it ideal for applications involving large rings.

Roadmap. Section 2 provides a technical overview of our results. Section 3 recalls some preliminaries. Section 4 presents our SMA scheme in detail and analyzes its security. Section 5 describes our scoped linkable ring signature framework and its instantiation, along with the corresponding security proofs. Section 6 evaluates the performance of our instantiation. Finally, Section 7 concludes the paper.

2 Technical Overview

Our set membership argument SMA is based on the vector commitment construction from [31]. Using a type-3 bilinear group $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where the pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and a common reference string $\mathsf{pp} = \{(g_i := g^{\beta^i})_{i\in[0,2n]\setminus\{n+1\}}, (\bar{g}_i := \bar{g}^{\beta^i})_{i\in[0,n]}\}$. Here, the elements marked with a bar (e.g., $\bar{g}_i$) lie in $\mathbb{G}_2$. According to [31], committing to a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$ is done by sampling $r \leftarrow\!\$\, \mathbb{Z}_p$ and computing $C = g^r \cdot \prod_{j=1}^n g_j^{v_j} = g^{r+\sum_{j=1}^n v_j\cdot\beta^j}$.

To open a position $i \in [n]$ of $\boldsymbol{v}$, the committer reveals a proof

$$\pi_i = g_{n+1-i}^r \cdot \prod_{j=1, j\neq i}^{n} g_{n+1-i+j}^{v_j} = \left(C/g^{v_i\cdot\beta^i}\right)^{\beta^{n+1-i}}$$

which is verified by checking that $e(C, \bar{g}_{n+1-i}) = e(\pi_i, \bar{g})\cdot e(g_n, \bar{g}_1)^{v_i}$.

Notably, the commitment allows proving that a committed $\boldsymbol{v} \in \mathbb{Z}_p^n$ satisfies an inner product relation $\langle \boldsymbol{v}, \boldsymbol{t}\rangle = x$, for public $\boldsymbol{t} = (t_1, \ldots, t_n) \in \mathbb{Z}_p^n$ and $x \in \mathbb{Z}_p$ [19]. By raising the verification equation to the power $t_i \in \mathbb{Z}_p$ and taking the product over all $i \in [n]$, we obtain

$$e\left(C, \prod_{i=1}^{n} \bar{g}_{n+1-i}^{t_i}\right) = e\left(\prod_{i=1}^{n} \pi_i^{t_i}, \bar{g}\right) \cdot e\left(g_n, \bar{g}_1\right)^{\sum_{i=1}^n v_i\cdot t_i} \tag{1}$$

This allows us to aggregate multiple openings of $\boldsymbol{v}$ by setting the range of i to the opening positions, and assigning t_i to random values derived by hashing C.

Set Membership Argument (SMA). Let $\Phi := (\phi_1, \phi_2, \ldots, \phi_m) \in \mathbb{Z}_p^m$ be a set. Using the previously defined public parameters pp, which also determine n as the upper bound of the set size. Using the above observation, we now show how to construct our constant-size set membership argument SMA. Consider a Pedersen commitment [38] $C = g^r \cdot g_1^{\phi_\ell}$ for $\phi_\ell \in \Phi$ in group $\mathbb{G}_1$, where $\ell \in [1, m]$ is the index. Let $\boldsymbol{b} := (b_1, \cdots, b_n) \in \{0,1\}^n$ be a one-hot vector where the ℓ-th position is 1 and all other positions are 0. The prover generate an auxiliary commitment to $\boldsymbol{b}$ as $\bar{C}_b = \bar{g}^{r_b} \cdot \prod_{i=1}^n \bar{g}_i^{b_i} \in \mathbb{G}_2$. To prove that ϕ_ℓ is in Φ, the prover proves that the committed ϕ_ℓ and $\boldsymbol{b}$ satisfies: (a) $\langle \boldsymbol{b}, \Phi \parallel \boldsymbol{0}^{n-m}\rangle = \phi_\ell$; (b) $b_i \in \{0,1\}$ for each $i \in [n]$; (c) $\boldsymbol{b}$ contains a single 1 in its front m positions, i.e., $\langle \boldsymbol{b}, \boldsymbol{1}^m \parallel \boldsymbol{0}^{n-m}\rangle = 1$.

In order to prove (a), the prover can adapt (1) and generate a short proof $\prod_{i=1}^m \pi_i^{\phi_i} \in \mathbb{G}_1$ such that

$$e\left(\prod_{i=1}^{m} g_{n+1-i}^{\phi_i}, \bar{C}_b\right) = e\left(\prod_{i=1}^{m} \pi_i^{\phi_i}, \bar{g}\right) \cdot e(g_n, \bar{g}_1)^{\sum_{i=1}^m b_i\cdot\phi_i} \tag{2}$$

and show that the exponent above $e(g_n, \bar{g}_1)$ in (2) is equal to the committed ϕ_ℓ in $C = g^r \cdot g_1^{\phi_\ell}$. Since $\boldsymbol{b}$ satisfies $\sum_{i=1}^m b_i \cdot \phi_i = \phi_\ell$, and C satisfies $e(C, \bar{g}_n) = e(g_n, \bar{g}_1)^{\phi_\ell} \cdot e(g_n^r, \bar{g})$, the prover can actually compute $\pi_\phi = \prod_{i=1}^m \pi_i^{\phi_i}/g_n^r$ such that

$$e\left(\prod_{i=1}^{m} g_{n+1-i}^{\phi_i}, \bar{C}_b\right)/e(C, \bar{g}_n) = e(\pi_\phi, \bar{g}) \tag{3}$$

The proof of (c) can be completed using a similar idea as the proof of (a) in (2). The only difference is that ϕ_i ($i \in [m]$) in (2) is replaced by 1, and $\sum_{i=1}^m b_i \cdot \phi_i$ is also replaced by 1. Therefore, the proof is given by $\pi_o = \prod_{i=1}^m \pi_i$ which satisfies

$$e\left(\prod_{i=1}^{m} g_{n+1-i}, \bar{C}_b\right)/e(g_n, \bar{g}_1) = e(\pi_o, \bar{g}) \tag{4}$$

Note that we do not need to prove that the $n-m$ last positions of $\boldsymbol{b}$ are zeroes since the inner product in the right-hand-side member of (2) only involves the first m positions of $\boldsymbol{b}$.

The proof of (b) can be addressed using the binary proof of a committed vector $\boldsymbol{b}$ as described in [30], which leverages the aggregation properties of the vector commitment in [31] and the batching technique from Bulletproofs [8]. The final binary proof consists of an auxiliary commitment $C_s \in \mathbb{G}_1$ and $(\pi_{eq}, \pi_b) \in \mathbb{G}_1^2$.

To ensure knowledge soundness in the algebraic group model, we also need to prove that the commitment C involves only g and g_1. Since C is placed in the first source group, it must exclude $(g_2, \ldots, g_n)$ and $(g_{n+2}, \ldots, g_{2n})$ from C. To ensure $(g_2, \ldots, g_n)$ are

[1] A single group exponentiation is at least three orders of magnitude slower than a single field multiplication in our benchmarks and in [15].

Table 1: Comparison of various ring signature schemes.

Scheme	PP size	Proof Size	Verification Time	Proving Time	Setting	Model	Linkable
AOS [1]	$\mathcal{O}(1)$	$(n+1)\,\lvert\mathbb{Z}_p\rvert$	$n\,\mathbb{G}$ **Exps**	$n\,\mathbb{G}$ **Exps**	DL	RO	×
LWW [32]	$\mathcal{O}(1)$	$1\,\lvert\mathbb{G}\rvert + (n+1)\,\lvert\mathbb{Z}_p\rvert$	$2n\,\mathbb{G}$ **Exps**	$2n\,\mathbb{G}$ **Exps**	DL	RO	✓
DKNS [14]	$\mathcal{O}(1)$	$12\lvert\mathbb{Z}_N^*\rvert + 12\,\lvert\mathbb{Z}\rvert$	$n\,\mathbb{Z}_N^*$ **Exps**	$n\,\mathbb{Z}_N^*$ **Exps**	Strong RSA	RO	×
Nguyen [36]	$\mathcal{O}(n)$	$5\,\lvert\mathbb{G}_1\rvert + 2\,\lvert\mathbb{G}_T\rvert + 8\,\lvert\mathbb{Z}_p\rvert$	$n\,\mathbb{G}_1$ **Exps**	$n\,\mathbb{G}_1$ **Exps**	q-SDH	RO	×
BDR [7]	$\mathcal{O}(n)$	$50\,\lvert\mathbb{G}_1^*\rvert + 42\,\lvert\mathbb{G}_2^*\rvert + 3\,\lvert\mathbb{Z}_N\rvert$	$n\,\mathbb{G}_1^*$ **Exps**	$2n\,\mathbb{G}_1^*$ **Exps**	q-SDH+SXDH	CRS	×
GK [21]	$\mathcal{O}(1)$	$4\log n\,\lvert\mathbb{G}_1\rvert + (3\log n+1)\,\lvert\mathbb{Z}_p\rvert$	$(n+3\log n)\,\mathbb{G}$ **Exps**	$(n+2\log n)\,\mathbb{G}$ **Exps**	DL	RO	×
RingCT 3.0 [42]	$\mathcal{O}(n)$	$(2\log n+7)\lvert\mathbb{G}\rvert + 7\,\lvert\mathbb{Z}_p\rvert$	$(2n+2\log n)\,\mathbb{G}$ **Exps**	$10n\,\mathbb{G}$ **Exps**	DL+q-DDHI	RO	✓
Omniring [29]	$\mathcal{O}(n)$	$\big(2\log(n+2)+4\big)\,\lvert\mathbb{G}\rvert + 5\,\lvert\mathbb{Z}_p\rvert$	$(2n+2\log n)\,\mathbb{G}$ **Exps**	$10n\,\mathbb{G}$ **Exps**	DL+DDH	RO	✓
CLSAG [18]	$\mathcal{O}(n)$	$3\,\lvert\mathbb{G}_1\rvert + (n+1)\,\lvert\mathbb{Z}_p\rvert$	$5n\,\mathbb{G}$ **Exps**	$3n\,\mathbb{G}$ **Exps**	DL+DDH	RO	✓
DualRing-EC [41]	$\mathcal{O}(n)$	$(2\log n+1)\,\lvert\mathbb{G}\rvert + 3\,\lvert\mathbb{Z}_p\rvert$	$n\,\mathbb{G}$ **Exps**	$4n\,\mathbb{G}$ **Exps**	DL	RO	×
DualDory [6]	$\mathcal{O}(n)$	$6\log n\,\lvert\mathbb{G}_T\rvert + 7\,\lvert\mathbb{G}_1\rvert + 2\,\lvert\mathbb{G}_2\rvert$	$10\log n\,\mathbb{G}_T$ **Exps** (PreComp[b]: $3n\,\mathbb{P}$)	$10n\,\mathbb{P} + 5n\,\mathbb{G}$ **Exps**	SXDH	RO	×[a]
LLRing-DL [24]	$\mathcal{O}(n)$	$\big(4\log n+9\big)\,\lvert\mathbb{G}\rvert + 7\,\lvert\mathbb{Z}_p\rvert$	$n\,\mathbb{G}$ **Exps** (PreComp[b]: $n\,\mathbb{G}$ **Exps**)	$10n\,\mathbb{G}$ **Exps**	DL	RO	✓
LLRing-P [24]	$\mathcal{O}(n)$	$\big(6\log n+10\big)\,\lvert\mathbb{G}_T\rvert + 11\,\lvert\mathbb{Z}_p\rvert$	$10\log n\,\mathbb{G}_T$ **Exps** (PreComp[b]: $3n\,\mathbb{P} + n\,\mathbb{G}_1$ **Exps**)	$10n\mathbb{P} + 2n\mathbb{G}_1$ **Exps** $+2n\mathbb{G}_2$ **Exps**	SXDH	RO	✓
Ours	$\mathcal{O}(n)$	$10\,\lvert\mathbb{G}_1\rvert + 5\,\lvert\mathbb{G}_2\rvert + 1\,\lvert\mathbb{Z}_p\rvert$	$5\mathbb{G}_1$ **Exps** $+ 7\mathbb{G}_2$ **Exps** $+17\,\mathbb{P} + 3n\,\mathbb{Z}_p$ **Muls**	$4n\,\mathbb{G}_1$ **Exps** $+ 2n\,\mathbb{G}_2$ **Exps**	(m, n)-DL + q-SDH	CRS + RO + AGM	✓

Note: The comparison highlights only the most significant terms influencing performance. The parameter n denotes the ring size. **PP size** refers to the size of the public parameters. Symbols such as $\lvert\mathbb{G}\rvert$ represent group elements; $\lvert\mathbb{G}_1\rvert$, $\lvert\mathbb{G}_2\rvert$, $\lvert\mathbb{G}_T\rvert$, and $\lvert\mathbb{Z}_p\rvert$ refer to elements in the first source group, second source group, target group, and the field of prime order p, respectively; $\lvert\mathbb{G}_1^*\rvert$ and $\lvert\mathbb{G}_2^*\rvert$ denote elements in the first and second source groups of composite order N, while $\lvert\mathbb{Z}_N\rvert$ represents field elements of composite order N; $\lvert\mathbb{Z}_N^*\rvert$ and $\lvert\mathbb{Z}\rvert$ correspond to RSA group elements and integers, respectively. **Exps** represent group exponentiations, **Muls** represent field multiplications, and $\mathbb{P}$ denotes pairing operations. [a] [6] identifies a vulnerability in the linkability by [24]. [b] Precomputations by the verifier can only occur if the ring is known in advance, prior to the signer generating the ring signature, which implies the use of a static ring.

Table 2: Signature size comparison (Bytes).

Scheme	Number of users											
	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
AOS [1]	288	544	1056	2080	4128	8224	16416	32800	65568	131104	262176	524160
LWW [32]	321	577	1089	2113	4161	8257	16449	32833	65501	131137	262209	524193
DKNS [14]	3840											
Nguyen [36]	1189											
BDR [7]	35520											
GK [21]	716	944	1172	1400	1628	1856	2084	2312	2540	2768	2996	3224
RingCT 3.0 [42]	653	719	785	851	917	983	1049	1115	1181	1247	1313	1379
Omniring [29]	556	622	688	754	820	886	952	1018	1084	1150	1216	1282
CLSAG [18]	387	643	1155	2179	4227	8323	16515	32899	65667	131203	262275	524259
DualRing-EC [41]	327	393	459	525	591	657	723	789	855	921	987	1053
DualDory [6]	7273	9577	11881	14185	16489	18793	21097	23401	25705	28009	30313	32617
LLRing-DL [24]	917	1049	1181	1313	1445	1577	1709	1841	1973	2105	2237	2369
LLRing-P [24]	11104	13408	15712	18016	20320	22624	24928	27232	29536	31840	34144	36448
Ours	**687**											

excluded, we verify that their exponents in C opens to zero using the aggregation technique in (1). For $(g_{n+2}, \ldots, g_{2n})$, since their orders exceed $n+1$, the aggregation technique cannot be directly applied. Instead, we shift the order of C by $n-1$, since a valid C always satisfies:

$$e(C, \bar{g}_{n-1}) = e(\pi_{d2}, \bar{g}), \tag{5}$$

where $\pi_{d2} = g_{n-1}^{r} g_n^{\phi_\ell}$. The absence of $(g_{2n+1}, \ldots, g_{3n-1})$ in the public parameters and the hardness of the $(2n, n)$-dlog assumption ensure soundness.

To minimize the proof size, we leverage the linearity of the above verification equations to aggregate all proofs into a single element $\pi \in \mathbb{G}_1$. The final proof of SMA is $\boldsymbol{\pi} = (\bar{C}_b, C_s, \pi)$, which consists of the commitment $\bar{C}_b \in \mathbb{G}_2$ to the one-hot binary vector $\boldsymbol{b}$, the auxiliary commitment $C_s \in \mathbb{G}_1$ for the binary proof of $\boldsymbol{b}$, and the aggregated proof $\pi \in \mathbb{G}_1$.

Reduce Verifier Computation of SMA. In SMA, the verifier must compute several multi-exponentiations, such as $\prod_{i=1}^{m} g_{n+1-i}^{\phi_i}$ in (3), and this remains the case after proof aggregation. By exploiting the fact that the public parameters pp have a structure similar to that of KZG polynomial commitments [26], a recently noted technique [9] can be used to outsource these computations to the prover, allowing the verifier to perform only a constant number of expensive operations.

For simplicity, we illustrate the process of outsourcing multi-exponentiations using $\prod_{i=1}^{m} g_{n+1-i}^{\phi_i}$ as an example. Let $f(X) = \sum_{i=1}^{m} \phi_i \cdot X^{n+1-i}$. The outsourcing of $\prod_{i=1}^{m} g_{n+1-i}^{\phi_i} \in \mathbb{G}_1$ proceeds as follows. The prover first computes $F = g^{f(\beta)} = \prod_{i=1}^{m} g_{n+1-i}^{\phi_i} \in \mathbb{G}_1$ and subsequently provides the proof of the opening of $f(X)$ at $z \in \mathbb{Z}_p$, denoted as $\pi_{\mathsf{kzg}} = g^{(f(\beta)-f(z))/(\beta-z)} \in \mathbb{G}_1$, where z is randomly sampled by hashing F. The prover then provides F and π_{kzg} to the verifier. To verify that F is well-formed, the verifier evaluates $f(z)$ and checks if

$$e(F \cdot g^{-f(z)}, \bar{g}) = e(\pi_{\mathsf{kzg}}, \bar{g}_1 \cdot \bar{g}^{-z}). \tag{6}$$

By outsourcing, the verifier reduces its workload to a constant number of exponentiations and pairings, along with $\mathcal{O}(n)$ operations in $\mathbb{Z}_p$ for evaluating $f(z)$. This idea can also be extended to outsourcing the computation of multi-exponentiations in $\mathbb{G}_2$.

In the final verification equation (14), the verifier needs to outsource three polynomial multi-exponentiation computations: one

in $\mathbb{G}_1$ and two in $\mathbb{G}_2$. By leveraging the linearity of the verification equation (6), the proofs for these computations can also be aggregated into a single element in $\mathbb{G}_1$. Consequently, outsourcing introduces 3 additional computation results in $\mathbb{G}_1 \times \mathbb{G}_2^2$ and 1 aggregated proof in $\mathbb{G}_1$.

After outsourcing the verifier's computation, the complete proof for our set membership argument SMA consists of 4 elements in $\mathbb{G}_1$ and 3 elements in $\mathbb{G}_2$.

Scoped Linkable Ring Signature Framework. Our linkable ring signature (LRS) construction is compatible with any PKI-based cryptosystem. Let $(\mathsf{pk}', \mathsf{sk})$ be an arbitrary key pair within a PKI, satisfying $\mathsf{pk}' = \mathcal{M}(\mathsf{sk})$ for a public mapping $\mathcal{M}$. The public key pk of our LRS is defined as (pk', ϕ), where $\phi = \mathcal{H}_\phi(\mathsf{pk}') \in \mathbb{Z}_p$ is derived by hashing the public key pk'. We retain sk as the private key for our LRS. The ring $\mathsf{R} := (\phi_1, \ldots, \phi_m)$ is a publicly list of hashed public keys, and we require that each ϕ_i within the ring is derived using the same mapping function.

We build our LRS in a modular fashion. Given a ring $\mathsf{R} := (\phi_1, \ldots, \phi_m)$ and a scoped string sc, to sign a message m, the signer first generates a set membership proof for their public key element ϕ_ℓ in R. This can be achieved using our SMA[2]. Specifically, the signer computes a Pedersen commitment C to ϕ_ℓ, and then invokes the Prove algorithm of SMA to complete the proof. This procedure conceals any information about ϕ_ℓ through the perfect hiding property of the commitment C and the zero-knowledge property of SMA. Meanwhile, the signer computes the linking element L using their private key sk and the scoped string sc, i.e., $L = \mathcal{H}_L(\mathsf{sc}, \mathsf{sk})$, where L depends solely on the private key and the scoped string, thereby ensuring scoped linkability.

Note that generating a set membership proof for a public key is trivial because the ring R is publicly known. In fact, anyone can construct such a proof that passes verification. Consequently, we rely on ccΠ, a commit-carry zkSNARK, to prove that $L = \mathcal{H}_L(\mathsf{sc}, \mathsf{sk}_\ell)$ and $\phi_\ell^* = \mathcal{H}_\phi(\mathcal{M}(\mathsf{sk}_\ell))$ are correctly computed, given the public input (L, sc) and the witness $(\phi_\ell^*, \mathsf{sk}_\ell)$. ccΠ produces an additional commitment C' to ϕ_ℓ^* along with a proof π_{cc}. Although the ccΠ proof and the SMA proof are both generated, they do not guarantee that C and C' commit to the same underlying value. In other words, there is still no confirmation for the verifier that $\phi_\ell = \phi_\ell^*$. Therefore, the signer additionally invokes lkΠ, a commit-and-prove SNARK proving that one commitment and another set of commitments indeed commit to the same value, thus ensuring the consistency. We emphasize that the statements proven by ccΠ are fixed and independent of the ring, depending solely on the mapping $\mathcal{M}$ and hash functions $\mathcal{H}_\phi, \mathcal{H}_L$. Similarly, the statements proven by lkΠ are also independent of the ring.

At this point, the proof generated by the signer confirms their membership in the ring and the possession of the corresponding private key required to sign. However, it does not yet sign the message m. To address this, we extend the syntax of SMA, allowing its proof and verification algorithms to accept an additional input, a tag (which, in this case, is the message m), to provide randomness for the proof generation process[3]. This step completes the signature for m. The final LRS signature is given by

$$\sigma := (C, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L).$$

To verify the signature σ for a message m with respect to the ring R and a scoped string sc, the verifier independently validates all included proofs and accepts the signature only if all verifications succeed. Linkability is established by checking whether two signatures share the same linking element L. If L matches, it indicates that the two signatures were generated by the same signer within the same scope.

Instantiation and Efficiency. We instantiate ccΠ using ccGro16, the commit-carry variant of the Groth scheme [4, 10], and lkΠ using $\mathsf{CP}_{\mathsf{link}}$ [27]. Due to the modular design of our LRS construction, where the relations to be proven by ccΠ and lkΠ are fixed and independent of the ring, their public parameters can be shared across different signings[4]. This allows for the efficient instantiation of our LRS with ccGro16 and $\mathsf{CP}_{\mathsf{link}}$ while avoiding the need to regenerate their CRS.

Our LRS achieves a constant signature size and efficient verification even for large ad-hoc rings. Specifically, the final signature consists of $10\,\mathbb{G}_1$, $5\,\mathbb{G}_2$, $1\,\mathbb{Z}_p$. Using $BN254$, our signature is only 687 bytes. We have compiled the signature sizes of existing ring signatures, as shown in Table 2. To the best of our knowledge, this is the smallest among existing constant-size ring signatures, and once the ring size exceeds 32, it remains the smallest linkable ring signature overall.

Our LRS verifier performs only a constant number of expensive operations and a linear number of field operations in $\mathbb{Z}_p$. This property is derived from the SMA, which features constant expensive computations and linear inexpensive operations during verification, as well as from the constant verification of SNARKs. In scenarios with ad hoc rings, the verifier incurs only $\mathcal{O}(n)$ operations in $\mathbb{Z}_p$, whereas existing ring signatures often require $\mathcal{O}(n)$ group exponentiations [6, 7, 24]. We emphasize that operations in $\mathbb{Z}_p$ can typically be implemented several orders of magnitude faster than group exponentiations and pairing operation, leading to a notable speedup. Consequently, our design achieves significant performance gains, especially for large ad hoc rings.

3 Preliminary

Bilinear Groups. A bilinear group is given by $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where p is a prime, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order p. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient, non-degenerate pairing. We use a *type-3* setting, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$ with no known efficient isomorphism between them. Let $g \in \mathbb{G}_1$ and $\bar{g} \in \mathbb{G}_2$ be generators.
Negligible Functions. A function $f(n)$ is *negligible* if, for any constant $c > 0$, there exists N such that for all $n > N$, $f(n) < \frac{1}{n^c}$.

[2]The SMA requires a predefined upper bound n and $n \geq m$, which need be initialized through a trusted setup. While this introduces some limitations, it is often practical in many applications. For example, in privacy-preserving cryptocurrencies such as Monero, providing limited anonymity[28, 35] is a reasonable trade-off to maintain overall system performance. Moreover, our SMA provides security for arbitrary $m \leq n$, meaning the LRS framework only requires a single trusted up to generate and share the public parameters. In practice, we typically set $n = 2^k$ as the anonymity upper bound, allowing ring sizes between $2^{k-1}+1$ and 2^k to achieve strong anonymity while maintaining high efficiency.

[3]The concept of deriving randomness for a signature from the message itself has also been explored in prior works [21, 25].

[4]Public parameters can be shared provided that: (1) the same key generation algorithm instance, i.e., the same mapping $\mathcal{M}$, and hash functions $\mathcal{H}_L$ and $\mathcal{H}_\phi$ are used; (2) the actual ring size does not exceed the predefined upper bound.

Sampling and Algorithms. We write $x \leftarrow\!\$\ S$ to denote sampling x uniformly at random from a non-empty set S. All algorithms are assumed to be (probabilistic) polynomial-time, denoted by PPT. For a probabilistic algorithm A, $y \leftarrow A(x;r)$ means y is the output on input x with randomness r.

Vector Notation. Vectors are in bold, e.g. $\boldsymbol{a} = (a_1, \ldots, a_l)$. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, their inner product is $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \sum_{i=1}^n a_i b_i$. For $\boldsymbol{h} \in \mathbb{G}_b^n$ where $b \in \{0,1\}$, define $\langle \boldsymbol{a}, \boldsymbol{h} \rangle := \prod_{i=1}^n h_i^{a_i}$.

3.1 Cryptographic Assumptions

Our protocol is proven secure in both the *random oracle model* and the *algebraic group model* of Fuchsbauer *et al.* [16]. For any input $x \in \{0,1\}^*$, the random oracle $\mathcal{H}$ returns a random k-bit string $\mathcal{H}(x)$. Distinct inputs produce distinct outputs, while repeated queries with the same input yield the same output. In AGM model, an *algebraic* adversary $\mathcal{A}$ can only output group elements as exponentiations of previously seen elements. That is, any $y \in \mathbb{G}_1$ produced by $\mathcal{A}$ has the form $y = \prod_{i=1}^l x_i^{a_i}$ for previously observed x_i, and $\mathcal{A}$ must also output the exponent vector $(a_1, \ldots, a_l)$. This extends to asymmetric bilinear groups with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Our set membership argument relies on the hardness of the (m,n)-DLOG assumption [30], which generalizes the n-DLOG assumption to the bilinear setting.

Definition 3.1. (m,n)-**dlog** [30]: Let $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \bar{g}, e)$ be a bilinear group, g and $\bar{g}$ are randomly chosen generators. For integers m, n, given $(g, g^\beta, g^{\beta^2}, \ldots, g^{\beta^m}, \bar{g}, \bar{g}^\beta, \bar{g}^{\beta^2}, \ldots, \bar{g}^{\beta^n})$, where $\beta \leftarrow\!\$\ \mathbb{Z}_p$, to compute $\beta \in \mathbb{Z}_p$.

3.2 Non-interactive Arguments

A non-interactive zero-knowledge argument (NIZK) is a protocol between a prover and a verifier, both of which are PPT entities. It consists of three algorithms: (PP-Gen, Prove, Verify).

The PP-Gen algorithm takes as input a security parameter $\lambda \in \mathbb{N}$, and possibly a language-dependent parameter lp. It outputs the public parameter pp and a trapdoor τ. Let $\mathcal{R}$ denote the non-deterministic polynomial (NP) relation to be proven. We allow pp to parameterize the relation (e.g., when pp specifies the commitment key of a commitment scheme). This leads to a parameterized relation $\mathcal{R}_{\mathsf{pp}}$ such that $\mathcal{R}_{\mathsf{pp}}(x,w) = 1 \iff (x,w) \in \mathcal{R}_{\mathsf{pp}}$. We call x a statement and w a witness, and use $\mathcal{L}_{\mathcal{R}_{\mathsf{pp}}}$ to denote the language consisting of all true statements of the relation $\mathcal{R}_{\mathsf{pp}}$, i.e., $\mathcal{L}_{\mathcal{R}_{\mathsf{pp}}} = \{x \mid \exists w : (x,w) \in \mathcal{R}_{\mathsf{pp}}\}$.

The algorithms Prove and Verify both take as inputs the public parameter pp and a statement x. The Prove algorithm also takes a secret witness w as input and outputs a proof π if $(x,w) \in \mathcal{R}_{\mathsf{pp}}$. The Verify algorithm, given the proof π, determines its validity and outputs 1 (accept) if the proof is valid, or 0 (reject) otherwise.

A NIZK must satisfy *completeness* and two key security properties: *zero-knowledge* and *knowledge soundness*.

Completeness. For any security parameter $\lambda \in \mathbb{N}$ and any instance sampler $\mathcal{S}$ outputs $(x,w) \leftarrow\!\$\ \mathcal{R}_{\mathsf{pp}}$, there is:

$$\Pr\left[\begin{array}{c} (\mathsf{pp},\tau) \leftarrow \mathsf{PP\text{-}Gen}(1^\lambda, \mathsf{lp}) \\ (x,w) \leftarrow \mathcal{S}(\mathsf{pp}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w) \end{array} : \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1\right] = 1$$

Zero-knowledge. For any $\lambda \in \mathbb{N}$ and any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a PPT simulator Sim such that

$$\Pr\left[\begin{array}{c} (\mathsf{pp},\tau) \leftarrow \mathsf{PP\text{-}Gen}(1^\lambda, \mathsf{lp}) \\ (x,w) \leftarrow \mathcal{A}_1(\mathsf{pp}), b \leftarrow\!\$\ \{0,1\} \\ \pi_0 \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w), \pi_1 \leftarrow \mathsf{Sim}(\mathsf{pp}, x, \tau) \\ b' = \mathcal{A}_2(\mathsf{pp}, x, \pi_b) \end{array} : b = b'\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $(x,w) \in \mathcal{R}_{\mathsf{pp}}$ and b is a random coin.

Knowledge Soundness. For any $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_{\mathcal{A}}$ that possesses the ability to program the random oracle $\mathcal{O}$ such that

$$\Pr\left[\begin{array}{c} (\mathsf{pp},\tau) \leftarrow \mathsf{PP\text{-}Gen}(1^\lambda, \mathsf{lp}) \\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp}, x, \pi) \end{array} : \begin{array}{c} \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1 \\ \wedge (x,w) \notin \mathcal{R}_{\mathsf{pp}} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Knowledge soundness requires that if an adversary can generate a valid proof for a statement x, then there exists an extractor that can extract a witness w such that $(x,w) \in \mathcal{R}_{\mathsf{pp}}$.

Here we also consider *simulation extractability*, a strengthening of knowledge soundness, which stipulates that an adversary cannot successfully prove a false statement, even after observing simulated proofs of statements corresponding to instances x of its choice.

Simulation Extractability. For any $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$, which possesses the ability to program the random oracle $\mathcal{O}$ and the simulation oracle Sim, such that

$$\Pr\left[\begin{array}{c} (\mathsf{pp},\tau) \leftarrow \mathsf{PP\text{-}Gen}(1^\lambda, \mathsf{lp}) \\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O},\mathsf{Sim}(\cdot,\cdot,\tau)}(\mathsf{pp}) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp}, x, \pi) \end{array} : \begin{array}{c} \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1 \\ \wedge (x,w) \notin \mathcal{R}_{\mathsf{pp}} \\ \wedge (x,\pi) \notin Q \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where Q is the set of simulated statement-proof pairs generated by adversary's queries to the simulation oracle Sim.

We extend the SMA syntax in Section 4 by adding a Com algorithm. Given the public parameter pp and a vector $\boldsymbol{x} \in D^n$, Com outputs a commitment C used in the relation $\mathcal{R}_{\mathsf{pp}}$. This extension ensures that the NIZK has the commit-carry property, as described in Section 3.4.

3.3 Commitment Schemes

We briefly review the concept of non-interactive commitment schemes.

Definition 3.2. A commitment scheme Com is a tuple of algorithms (Setup, CM, VerC) as defined below:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$: Given a security parameter, the algorithm outputs a commitment key ck.
- $\mathsf{CM}(\mathsf{ck}, z) \to (C, o)$: Commits message z, outputs a commitment C and an opening o.
- $\mathsf{VerC}(\mathsf{ck}, C, z, o) \to b$: Given ck, a commitment C, a message z, and an opening o, verifies if C is a valid commitment for z and outputs $b = 1$ for *accept* or $b = 0$ for *reject*.

A commitment scheme must satisfy both *binding* and *hiding*. *Hiding* ensures that the committed value z cannot be determined from the commitment C, while *binding* guarantees that C can be opened to only one value. In a perfectly hiding scheme, even an adversary with unlimited computational power cannot reveal the committed value from C.

Pedersen Vector Commitment (PVC). We recall the extended Pedersen commitment scheme for vectors, which is computationally

binding and perfectly hiding. PVC.Setup additionally inputs $n \in \mathbb{N}$, which specifies the length of the committed vector. Let $\boldsymbol{w} \in \mathbb{Z}_p^n$, PVC works as follows.

- PVC.Setup$(1^\lambda, n)$: Output $\boldsymbol{h} \leftarrow\!\$\ \mathbb{G}_p^{n+1}$.
- PVC.CM$(\boldsymbol{h}, \boldsymbol{w})$: Return $(C, o) := (\langle (o, \boldsymbol{w}), \boldsymbol{h} \rangle, o)$ for $o \leftarrow\!\$\ \mathbb{Z}_p$.
- PVC.VerC$(\boldsymbol{h}, C, \boldsymbol{w}, o)$: Output 1 iff $C = \langle (o, \boldsymbol{w}), \boldsymbol{h} \rangle$.

When $n = 1$, the Pedersen Vector Commitment reduces to the Pedersen Commitment.

KZG Polynomial Commitment [26]. Polynomial commitment schemes include two additional algorithms: Open and VerOpen. The Setup algorithm of KZG additionally takes a parameter n, which specifies the upper bound on the degree of the polynomial. It outputs the commitment key $\mathsf{ck} = \{g_i := g^{\beta^i}\}_{i=0}^{n-1}$ and the verification key $\mathsf{vk} = \bar{g}^\beta$, allowing commitments to polynomials of degree less than n. For a polynomial $f(X) = \sum_{i=0}^{n-1} a_i X^i$, the commitment is computed as $C := \prod a_i g_i \leftarrow \mathsf{KZG.CM}(\mathcal{G}, \mathsf{ck}, f)$.

To open a point (z, y), where $y = f(z)$, KZG uses the polynomial remainder theorem, which states that $f(z) = y \iff \exists q(X) : f(X) - y = q(X)(X - z)$. The proof is a KZG commitment to the quotient polynomial $q(X)$. Let $q(X) = \sum_{i=0}^{n-1} b_i X^i$, then the proof $Q := \prod b_i g_i \leftarrow \mathsf{KZG.Open}(\mathcal{G}, \mathsf{ck}, f, z)$. Using the verification key, the verifier runs $\mathsf{KZG.VerOpen}(\mathcal{G}, \mathsf{vk}, C, Q, z, y)$, checking whether $e(C \cdot g^{-y}, \bar{g}) = e(Q, \mathsf{vk} \cdot \bar{g}^{-z})$.

3.4 zkSNARK Components

zero-knowledge Succinct Non-interactive Arguments of Knowledge (zkSNARKs) are characterized by short and efficiently verifiable proofs [10, 34], which is why they are referred to as succinct. We recall two specific zkSNARKs components introduced by Campanelli *et al.* [10]: the commit-carrying zkSNARK (cc-SNARK), which applies to any NP relation, and $\mathsf{CP}_{\mathsf{link}}$, which targets the specific relation $\mathcal{R}_{\mathsf{link}}$.

cc-SNARKs. Intuitively, a cc-SNARK extends the standard zk-SNARK by including a commitment to a portion of the witness in the proof. The witness w is partitioned into two parts, u and v, where u represents the portion that is committed to in the proof and is thus referred to as the *committed portion*.

Definition 3.3. A cc-SNARKs [10] for an NP relation family $\mathcal{R}_\lambda$, where $\lambda \in \mathbb{N}$, is a tuple of algorithms $\mathsf{cc}\Pi := (\mathsf{KG}, \mathsf{Prv}, \mathsf{Vrf}, \mathsf{VerC})$ as follows:

- $\mathsf{KG}(1^\lambda, \mathcal{R}) \to (\mathsf{pp}, \tau)$: Given a security parameter λ and a relation $\mathcal{R} \in \mathcal{R}_\lambda$, outputs the public parameters, including a commitment key ck, a evaluation key ek, a verification key vk. It also outputs a trapdoor τ.
- $\mathsf{Prv}(\mathsf{ek}, x, w) \to (C, \pi; o)$: Input ek, a statement x, and a valid witness $w = (u, v)$, outputs a proof π, a commitment C, and an opening o such that $\mathsf{VerC}(\mathsf{ck}, C, u, o) = 1$.
- $\mathsf{Vry}(\mathsf{vk}, x, C, \pi) \to b$: On input vk, a statement x, a commitment C, and a proof π, output $b = 1$ for *accept* or $b = 0$ for *reject*.
- $\mathsf{VerC}(\mathsf{ck}, C, u, o) \to b$: On input ck, a commitment C, a message u, and an opening o, outputs $b = 1$ (accept) if C opens to u, otherwise $b = 0$ (reject).

A cc-SNARK must satisfy *completeness*, *knowledge soundness*, *zero-knowledge*, *binding*, and *succinctness*. Its *completeness* and *zero-knowledge* properties are analogous to those in Section 3.2. For details, see Section 3.4 of [10].

Knowledge Soundness. A ccSNARK $\mathsf{cc}\Pi$ is knowledge sound if for any $\lambda \in \mathbb{N}$, $\mathcal{R} \in \mathcal{R}_\lambda$, and any PPT $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{c} (\mathsf{pp}, \tau) \leftarrow \mathsf{KG}(1^\lambda, \mathcal{R}) \\ (x, C, \pi) \leftarrow \mathcal{A}(\mathcal{R}, \mathsf{pp}) \\ (u, o, v) \leftarrow \mathcal{E}_{\mathcal{A}}(\mathcal{R}, \mathsf{pp}) \end{array} : \begin{array}{c} \mathsf{Vry}(\mathsf{vk}, x, C, \pi) = 1 \wedge \\ \left(\begin{array}{c} \mathsf{VerC}(\mathsf{ck}, C, u, o) = 0 \\ \vee (x, (u, v)) \notin \mathcal{R}_{\mathsf{pp}} \end{array} \right) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Binding. A ccSNARK $\mathsf{cc}\Pi$ is binding if for any polynomial-time adversary $\mathcal{A}$, there is

$$\Pr\left[\begin{array}{c} (\mathsf{pp}, \tau) \leftarrow \mathsf{KG}(1^\lambda, \mathcal{R}) \\ (C, u, o, u', o') \leftarrow \mathcal{A}(\mathcal{R}, \mathsf{pp}) \end{array} : \begin{array}{c} \mathsf{VerC}(\mathsf{ck}, C, u, o) = 1 \wedge \\ \mathsf{VerC}(\mathsf{ck}, C, u', o') = 1 \\ \wedge u \neq u' \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Succinctness. A ccSNARK $\mathsf{cc}\Pi$ is said succinct if the running time of Vry is $\mathsf{poly}(\lambda) \cdot (\lambda + |x| + \log |w|)$ and the size of the proof is $\mathsf{poly}(\lambda) \cdot (\lambda + \log |w|)$.

Here we also provide the *simulation extractability* for ccSNARK.

Simulation Extractability. A ccSNARK $\mathsf{cc}\Pi$ is simulation extractable if for any $\lambda \in \mathbb{N}$, $\mathcal{R} \in \mathcal{R}_\lambda$, and any PPT $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{c} (\mathsf{pp}, \tau) \leftarrow \mathsf{PP\text{-}Gen}(1^\lambda, \mathsf{lp}) \\ (x, C, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathcal{R}, \mathsf{pp}) \\ (u, o, v) \leftarrow \mathcal{E}_{\mathcal{A}}(\mathcal{R}, \mathsf{pp}) \end{array} : \begin{array}{c} \mathsf{Vry}(\mathsf{vk}, x, C, \pi) = 1 \\ \wedge (x, C, \pi) \notin Q \wedge \\ \left(\begin{array}{c} \mathsf{VerC}(\mathsf{ck}, C, u, o) = 0 \\ \vee (x, (u, v)) \notin \mathcal{R}_{\mathsf{pp}} \end{array} \right) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where Q is the set of simulated statement-proof pairs generated by adversary's queries to the simulation oracle Sim.

$\mathsf{CP}_{\mathsf{link}}$ [10, 27]. $\mathsf{CP}_{\mathsf{link}}$ is a SNARK for following relation $\mathcal{R}_{\mathsf{link}}$. Let C' be a single vector commitment, and $(C_1, \ldots, C_l)$ be another set of vector commitments using the same verification algorithm but different keys. $\mathsf{CP}_{\mathsf{link}}$ proves that these two sets open to the same vector, provided that the commitment algorithms for both sets follow the Pedersen verification.

Consider the case where the commitments are in the first source group $\mathbb{G}$. Let n_i denote the length of the vector committed to by C_i, where $n_i \leq n$ for $i \in [l]$ and $\sum n_i = m$. Let $\boldsymbol{h} \in \mathbb{G}^n$ represent the commitment key for C_i, and let $\boldsymbol{f} \in \mathbb{G}^m$ be the commitment key for C'. Let $\boldsymbol{u}_i$ and o_i denote the message vector and opening for C_i, respectively, and let o' be the opening for C'. The formal definition of $\mathcal{R}_{\mathsf{link}}$ is as follows:

$$\mathcal{R}_{\mathsf{link}} = \left\{ \begin{array}{c} \left(C', (C_i)_{i\in[l]}; (\boldsymbol{u}_i)_{i\in[l]}, (o_i)_{i\in[l]}, o'\right) : \\ \bigwedge_{i\in[l]} C_i = \langle (o_i, \boldsymbol{u}_i), \boldsymbol{h}_{[0,n_i]} \rangle \\ C' = \langle (o', \boldsymbol{u}_1, \cdots, \boldsymbol{u}_l), \boldsymbol{f} \rangle \end{array} \right\}$$

The syntax of $\mathsf{CP}_{\mathsf{link}}$ for relation $\mathcal{R}_{\mathsf{link}}$ is recalled below:

- $\mathsf{CP}_{\mathsf{link}}.\mathsf{KG}(1^\lambda, \mathcal{R}_{\mathsf{link}}, \mathsf{ck}', \mathsf{ck}) \to (\mathsf{ek}, \mathsf{vk})$: Parse $\mathsf{ck} := \boldsymbol{h} \in \mathbb{G}^{n+1}$ and $\mathsf{ck}' := \boldsymbol{f} \in \mathbb{G}^{m+1}$, return $(\mathsf{ek}, \mathsf{vk})$.
- $\mathsf{CP}_{\mathsf{link}}.\mathsf{Prv}(\mathsf{ek}, x, w)) \to \pi$: Parse $x := (C', (C_i)_{i\in[l]})$ and $w := ((\boldsymbol{u}_i)_{i\in[l]}, (o_i)_{i\in[l]}, o')$, return proof π.
- $\mathsf{CP}_{\mathsf{link}}.\mathsf{Vrf}(\mathsf{vk}, x, \pi)$: Parse $x := (C', (C_i)_{i\in[l]})$, output $b = 1$ for *accept* or $b = 0$ for *reject*.

$\mathsf{CP}_{\mathsf{link}}$ achieves *one-time simulation soundness* and *zero-knowledge*, as demonstrated in [27]. The prover performs a single multi-exponentiation

of length $4(m+l+1)$ and produces two $\mathbb{G}_1$ elements as the proof, while the verifier performs a single multi-exponentiation of length $l+1$ and $l+3$ pairings.

3.5 Scoped Linkable Ring Signatures

We review the definitions of scoped linkable ring signature in [23, 40], where two signatures link if and only if they are produced by the same signer under the same scope (e.g., context-specific information).

Definition 3.4. A scoped linkable ring signature scheme LRS consists of five PPT algorithms LRS := (Setup, KeyGen, Sign, Verify, Link) defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{pp}$: On input a security parameter λ and a ring size upper bound n, outputs the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Takes as input the public parameters pp and outputs a public-private key pair.
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m}, \mathsf{sc}, R) \to \sigma$: Takes as input the private key sk, message m, scoped string sc, and a ring R (a list of public keys that includes the public key corresponding to sk), and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{pp}, \sigma, \mathsf{m}, \mathsf{sc}, R) \to b$: Takes as input the signature σ, message m, scoped string sc, and ring R, and outputs a bit $b = 1$ for *accept* or $b = 0$ for *reject*.
- $\mathsf{Link}(\mathsf{pp}, \mathsf{sc}, \sigma, \sigma') \to b$: On input two valid linkable ring signatures under the same scope sc, outputs a bit $b = 1$ for *linked* or $b = 0$ for *unlinked*.

Correctness. A linkable ring signature scheme is correct if:

- For all $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $i \in [n]$, a ring $R := (\mathsf{pk}_1, \cdots, \mathsf{pk}_n)$, and all message strings m, the verification algorithm always accepts valid signatures: $\mathsf{Verify}(\mathsf{pp}, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m}, \mathsf{sc}, R), \mathsf{m}, \mathsf{sc}, R) = 1$.
- For any $(\mathsf{sc}, \sigma, \sigma')$ honestly generated using the same sk for the same scope string sc, the linking algorithm always correctly identifies them as linked: $\mathsf{Link}(\mathsf{pp}, \sigma, \sigma', \mathsf{sc}) = 1$.

Security. A LRS scheme should satisfy three essential properties: anonymity, scoped linkability, and non-slanderability. Notably, the property of unforgeability in LRS schemes can be derived from the combined guarantees of linkability and non-slanderability, as shown in [3]. Detailed security definitions are provided in Appendix B due to space limitations.

4 Set Membership Argument

In this section, we first present SMA, a zero-knowledge set membership argument with a constant number of group elements and $\mathcal{O}(n)$ exponentiations for both the prover and verifier. Then, we describe a technique based on KZG polynomials commitments [26], allowing the verifier to perform a constant number of expensive operations, such as exponentiations and pairings.

4.1 A Set Membership Argument

Let pp denote the public parameters, and let $\Phi := (\phi_1, \phi_2, \ldots, \phi_m) \in \mathbb{Z}_p^m$ be a set. Consider a Pedersen commitment $C = g^r \cdot g_1^{\phi_\ell}$ in the source group $\mathbb{G}_1$, constructed using parameters from pp for the element $\phi_\ell \in \Phi$, where $\ell \in [1, m]$ is the index. The set membership relation parameterized by the public parameter pp is formally defined as follows:

$$\mathcal{R}_{\mathsf{pp}}^{\mathsf{SMA}} = \{(\Phi, C; \phi_\ell, r) : C = g^r \cdot g_1^{\phi_\ell} \wedge \phi_\ell \in \Phi\}$$

where the commitment key (g, g_1) and the group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are specified by the public parameters pp.

Let $\boldsymbol{b} := (b_1, \cdots, b_n) \in \{0,1\}^n$ denote a one-hot vector where the ℓ-th position is 1 and all other positions are 0. The prover commits to $\boldsymbol{b}$ using a constant-size vector commitment $\bar{C}_b \in \mathbb{G}_2$. Let $\boldsymbol{v_1} \parallel \boldsymbol{v_2}$ denote the concatenation of two vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. To prove that ϕ_ℓ is in Φ, the prover will prove knowledge of an opening $\boldsymbol{b} \in \{0,1\}^n$ of $\bar{C}_b$, along with r, ϕ_ℓ such that $C = g^r \cdot g_1^{\phi_\ell}$ and

$$\langle \boldsymbol{b}, \Phi \parallel \boldsymbol{0}^{n-m} \rangle = \phi_\ell \wedge \langle \boldsymbol{b}, (\boldsymbol{b} - \boldsymbol{1}) \rangle = 0 \wedge \langle \boldsymbol{b}, \boldsymbol{1}^m \parallel \boldsymbol{0}^{n-m} \rangle = 1 \quad (7)$$

Our SMA is constructed following the relation described in (7). Specifically, in what follows, equations (8) and (9) jointly capture the fact that $\bar{C}_b$ commits to a binary vector $\boldsymbol{b}$ (i.e., $\langle \boldsymbol{b}, (\boldsymbol{b} - \boldsymbol{1}) \rangle = 0$). Building on this, equation (10) captures the fact that $\boldsymbol{b}$ is a one-hot vector containing a 1 in its first m positions (i.e., $\langle \boldsymbol{b}, \boldsymbol{1}^m \parallel \boldsymbol{0}^{n-m} \rangle = 1$). Finally, equation (13) is used to capture the fact that $\langle \boldsymbol{b}, \Phi \parallel \boldsymbol{0}^{n-m} \rangle = \phi_\ell$, i.e., $\phi_\ell \in \Phi$.

Moreover, to ensure knowledge soundness in the algebraic group model, we require (5) and (6) to capture that C is a commitment to the form $(\phi_\ell, 0, \cdots, 0) \in \mathbb{Z}_p^{2n}$. Specifically, (5) constrains the last $n-1$ positions to be zero, and (6) constrains positions $2 \sim n$ to be zero. Furthermore, due to the absence of the g_{n+1} term, these constraints together capture that C in the required form. We emphasize that while all components in our SMA rely on the $(2n, n)$-dlog assumption, equation (5) is distinct from other parts. Its soundness does not rely on the absence of g_{n+1} in pp but instead on the absence of $(g_{2n+1}, \cdots, g_{3n-1})$, a technique not previously noted in schemes based on [31]. This technique allows us to place the commitment to ϕ_ℓ in the first source group[5]. This makes our technique more versatile, as most schemes place commitments in the $\mathbb{G}_1$ rather than $\mathbb{G}_2$ for efficiency considerations.

Our SMA includes the following four probabilistic polynomial-time algorithms: PP-Gen, Com, Prove, and Verify.

PP-Gen(1^λ, 1^n): Input the security parameter λ, the upper bound n for the set size, and a type-3 bilinear group $\mathcal{G}$:

- Choose type-3 bilinear pairing group $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where l is a polynomial function and $p > 2^{l(\lambda)}$ is a prime. The groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order p, and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Sample generators $g \leftarrow\!\$\ \mathbb{G}_1$, $\bar{g} \leftarrow\!\$\ \mathbb{G}_2$.
- Compute $\mathsf{crs} = \{(g_i := g^{\beta^i})_{i \in [0,2n] \setminus \{n+1\}}, (\bar{g}_i := \bar{g}^{\beta^i})_{i \in [0,n]}\}$ as the common reference string, where the trapdoor $\beta \leftarrow\!\$\ \mathbb{Z}_p$.
- Choose hash functions $\boldsymbol{H} := (H_s, H_t, H_u, H_{\mathsf{agg}})$ that will be modeled as random oracles. Here, $H_s, H_t : \{0,1\}^* \to \mathbb{Z}_p^n$, $H_u : \{0,1\}^* \to \mathbb{Z}_p$, and $H_{\mathsf{agg}} : \{0,1\}^* \to \mathbb{Z}_p^6$.
- Let $\mathsf{ck} := (g, g_1)$ be the commitment key.
- Return public parameters $\mathsf{pp} := \{\mathsf{crs}, \mathsf{ck}, \boldsymbol{H}\}$.

[5] Although this technique is used in the context of set membership proofs, it can also be applied to range proofs, such as those in [30], avoiding the limitation of requiring the commitment $x \in [0, 2^n - 1]$ to be placed in the second source group $\mathbb{G}_2$.

Com(pp, ϕ_ℓ): Input $\phi_\ell \in \mathbb{Z}_p$, choose a random $r \leftarrow\!\$\, \mathbb{Z}_p$, and compute the Pedersen commitment $C = g^r \cdot g_1^{\phi_\ell}$. Return the commitment C and the opening information $r \in \mathbb{Z}_p$.

Prove(pp, (Φ, C), (ϕ_ℓ, r)): Input the set $\Phi := (\phi_1, \phi_2, \ldots, \phi_m)$, the Pedersen commitment $C = g^r \cdot g_1^{\phi_\ell}$ for $\phi_\ell \in \Phi$ with opening information $r \in \mathbb{Z}_p$. Let $\boldsymbol{b} \in \{0,1\}^m$ be a vector that is 1 at index $\ell \in [1, m]$ and 0 elsewhere (so that $\boldsymbol{b}$ satisfies the inner product relation $\langle \boldsymbol{b}, \Phi \rangle = \phi_\ell$).

(1) Let $(b_{m+1}, \ldots, b_n) = \mathbf{0}^{n-m}$. Choose $r_b \leftarrow\!\$\, \mathbb{Z}_p$ and compute $\bar{C}_b = \bar{g}^{r_b} \cdot \prod_{i=1}^m \bar{g}_i^{b_i} = \bar{g}^{r_b} \cdot \bar{g}_\ell$. Then, compute $\boldsymbol{s} := (s_1, \ldots, s_n) = H_s(C, \bar{C}_b)$. Choose $r_s \leftarrow\!\$\, \mathbb{Z}_p$ and compute $C_s = g^{r_s} \cdot \prod_{i=1}^m g_{n+1-i}^{s_i \cdot b_i} = g^{r_s} \cdot g_{n+1-\ell}^{s_\ell}$.

(2) Prove that C_s is a commitment to $(s_n \cdot b_n, \ldots, s_1 \cdot b_1)$. Specifically, let $\boldsymbol{t} := (t_1, \ldots, t_n) = H_t(\boldsymbol{s}, C, \bar{C}_b, C_s)$, and compute:

$$\pi_{eq} = \frac{\prod_{i=1}^n \left(g_{n+1-i}^{r_b} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j} \right)^{t_i \cdot s_i}}{\prod_{i=1}^n \left(g_i^{r_s} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1+i-j}^{s_j \cdot b_j} \right)^{t_i}}$$

which satisfies:

$$\frac{e\left(\prod_{i=1}^n g_{n+1-i}^{t_i \cdot s_i}, \bar{C}_b\right)}{e\left(C_s, \prod_{i=1}^n \bar{g}_i^{t_i}\right)} = e\left(\pi_{eq}, \bar{g}\right) \tag{8}$$

(3) Prove that $(b_1, \ldots, b_n) \in \{0,1\}^n$. This is achieved by verifying that for any random vector $\boldsymbol{s}$, it holds that $\sum_{i=1}^n s_i \cdot b_i \cdot (b_i - 1) = 0$. Specifically, compute π_b as follows:

$$\pi_b = g^{r_b r_s} \cdot \left(\prod_{j=1}^n g_{n+1-j}^{r_b s_j (b_j - 1)} \right) \cdot \prod_{i=1}^n \left(g_i^{b_i r_s} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-j+i}^{b_i s_j (b_j - 1)} \right)$$

which should satisfy the following equation:

$$e\left(C_s \cdot \prod_{j=1}^n g_{n+1-j}^{-s_j}, \bar{C}_b \right) = e\left(\pi_b, \bar{g}\right) \tag{9}$$

(4) Further prove that $(b_1, \ldots, b_m)$ is a one-hot vector, which is 1 at index $\ell \in [1, m]$ and 0 elsewhere. Specifically, compute:

$$\pi_o = \prod_{i=1}^m \left(g_{n+1-i}^r \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j} \right)$$

which should satisfy

$$\frac{e(\prod_{i=1}^m g_{n+1-i}, \bar{C}_b)}{e(g_n, \bar{g}_1)} = e\left(\pi_o, \bar{g}\right) \tag{10}$$

(5) Restrict the generation of C without using $(g_{n+2}, \ldots, g_{2n})$ from the crs. This can be achieved by shifting the order of C by $n-1$ times. Specifically, compute:

$$\pi_{d2} = g_{n-1}^r g_n^{\phi_\ell}$$

which should satisfy:

$$e(C, \bar{g}_{n-1}) = e(\pi_{d2}, \bar{g}) \tag{11}$$

(6) Further restrict the generation of C without using $(g_2, \ldots, g_n)$ from the crs, namely, prove that $C = g^r \cdot g_1^{\phi_\ell}$ opens to 0 in its 2 to n coordinates. Specifically, for each $i \in [2, n]$, compute $u_i = H_u(i, [2, n], C, \bar{C}_b, C_s) \in \mathbb{Z}_p$. Then, compute a aggregated proof π_{d1} as follows:

$$\pi_{d1} = \prod_{i=2}^n (g_{n+1-i}^r \cdot g_{n+2-i}^{\phi_\ell})^{u_i}$$

which should satisfy

$$e(C, \prod_{i=2}^n \bar{g}_{n+1-i}^{u_i}) = e(\pi_{d1}, \bar{g}) \tag{12}$$

(7) Check if $\phi_\ell \in \Phi$ by verifying whether $\langle \boldsymbol{b}, \Phi \rangle = \phi_\ell$. Specifically, compute:

$$\pi_\phi = g_n^{-r} \cdot \prod_{i=1}^m \left(g_{n+1-i}^{\phi_i \cdot r_b} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{\phi_i \cdot b_j} \right)$$

which satisfies

$$\frac{e(\prod_{i=1}^m g_{n+1-i}^{\phi_i}, \bar{C}_b)}{e(C, \bar{g}_n)} = e(\pi_\phi, \bar{g}) \tag{13}$$

(8) Compute $\boldsymbol{\delta} := (\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) = H_{\text{agg}}(C, \bar{C}_b, C_s) \in \mathbb{Z}_p^6$ and the aggregated proof:

$$\pi = \pi_{eq}^{\delta_{eq}} \cdot \pi_b^{\delta_b} \cdot \pi_o^{\delta_o} \cdot \pi_{d2}^{\delta_{d2}} \cdot \pi_{d1}^{\delta_{d1}} \cdot \pi_\phi^{\delta_\phi}$$

(9) Finally, output the final set membership proof $\boldsymbol{\pi} := (\bar{C}_b, C_s, \pi)$.

Note that we do not need to prove that the last $n - m$ positions of $\boldsymbol{b}$ are zero, because only the first m positions of $\boldsymbol{b}$ contribute to the exponent in the verification term $e(g_n, \bar{g}_1)$.

Verify(pp, (Φ, C), $\boldsymbol{\pi}$): Input the set $\Phi := (\phi_1, \phi_2, \ldots, \phi_m)$, the Pedersen commitment C, parse $(\bar{C}_b, C_s, \pi) := \boldsymbol{\pi}$.

(1) Compute all randomness as in Prove: $(s_1, \cdots, s_n) = H_s(C, \bar{C}_b)$, $(t_1, \cdots, t_n) = H_t(\boldsymbol{s}, C, \bar{C}_b, C_s)$, $u_i = H_u(i, [2, n], C, \bar{C}_b, C_s)$ for each $i \in [2, n]$, and $(\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) = H_{\text{agg}}(C, \bar{C}_b, C_s)$.

(2) Let $\delta_{o,i} = \delta_o$ and $\delta_{\phi,i} = \delta_\phi$ for $i \in [m]$. For $i \in [m+1, n]$, $\phi_i = \delta_{o,i} = \delta_{\phi,i} = 0$. Return 1 iff

$$\frac{e(C_s^{\delta_b} \cdot \prod_{i=1}^n g_{n+1-i}^{\delta_{o,i} + (\delta_{eq} \cdot t_i - \delta_b) \cdot s_i + \delta_{\phi,i} \cdot \phi_i}, \bar{C}_b)}{e(C, \bar{g}_n^{\delta_\phi} \bar{g}_{n-1}^{-\delta_{d2}} \prod_{i=2}^n \bar{g}_{n+1-i}^{-\delta_{d1} \cdot u_i}) \cdot e(C_s, \prod_{i=1}^n \bar{g}_i^{\delta_{eq} \cdot t_i}) \cdot e(g_n^{\delta_o}, \bar{g}_1)} = e(\pi, \bar{g}) \tag{14}$$

Correctness. The verification equation (14) is obtained by raising the sub-verification equations (8)-(13) to the powers δ_{eq}, δ_b, δ_o, δ_{d2}, δ_{d1}, δ_ϕ respectively, and then multiplying the results. Its correctness follows from the correctness of the sub-verification equations. We emphasize that none of the sub-proofs (e.g., π_{eq}) rely on the g_{n+1} term for an honest prover, and neither does the aggregated proof π. In Appendix A.1, we provide a detailed correctness proof for each of the above sub-verification equations.

Theorem 4.1. SMA *is statistical zero-knowledge in the ROM.*

Theorem 4.2. *Under the* $(2n, n)$*-DLOG assumption,* SMA *satisfies the simulation-extractability property in the ROM and the AGM.*

The proofs of Theorem 4.1 and Theorem 4.2 are formalized in Appendix A.2 and Appendix A.3, respectively.

Efficiency. The proof of our SMA involves only two elements from $\mathbb{G}_1$ and one element from $\mathbb{G}_2$. The prover's computation is dominated by $2n$ exponentiations in $\mathbb{G}_1$. Specifically, since $\boldsymbol{b}$ is a

one-hot vector, computing $\bar{C}_b$ requires a single exponentiation in $\mathbb{G}_2$, while C_s requires two exponentiations in $\mathbb{G}_1$. Additionally, the prover must compute π, which does not require the computation of all sub-proofs before aggregating them to obtain π. Instead, π can be directly expressed as $g^{P_\pi(\beta)}$, where $P_\pi[X]$ is a degree-$2n$ polynomial defined as

$$
\begin{aligned}
P_\pi[X] = &(r_b + \sum_{i=1}^{n} b_i \cdot X^{\ell}) \cdot \\
&(\delta_b r_s + \sum_{i=1}^{n} \left(\delta_b s_i b_i + \delta_{o,i} + (\delta_{eq} t_i - \delta_b) s_i + \delta_{\phi,i} \cdot \phi_i\right) X^{n+1-\ell}) \\
&- (r + \phi_\ell X) \cdot (\delta_\phi X^n - \delta_{d2} X^{n-1} - \sum_{i=2}^{n} \delta_{d1} u_i X^{n+1-i}) \\
&- (\sum_{i=1}^{n} \delta_{eq} t_i X^n) \cdot (r_s + \sum_{i=1}^{n} b_i s_i \cdot X^{n+1-\ell}) - X^{n+1} = \sum_{i=0}^{2n} \theta_i \cdot X^i
\end{aligned}
$$

The randomness $\boldsymbol{s}, \boldsymbol{t}, \{u_i\}_{i\in[2,n]}$ and the aggregation coefficients $\boldsymbol{\delta}$ are solely dependent on the commitments $(C, \bar{C}_b, C_s)$, which allows the prover can recover the coefficients of the polynomial $P_\pi[X]$. Furthermore, since $b_i = 0$ for all $i \in [n] \setminus \{\ell\}$, the polynomials marked in blue will reduce to only two terms, meaning that the prover requires just $O(n)$ multiplications to recover $P_\pi[X]$. Once the polynomial is obtained, the prover performs $2n$ exponentiations in $\mathbb{G}_1$ to compute π. In summary, the prover's computational cost consists of $2n+2$ exponentiations in $\mathbb{G}_1$, one exponentiations in $\mathbb{G}_2$, and $O(n)$ multiplications in $\mathbb{Z}_p$. On the other hand, the verifier's computation is dominated by $n+1$ exponentiations in $\mathbb{G}_1$, $2n+2$ exponentiations in $\mathbb{G}_2$, and 5 pairings.

Remark 1. *We emphasize that, we can extend the syntax of* Prove *and* Verify *to enhance functionality by allowing both algorithms to accept an additional tag $\mathcal{T}$ (i.e.,* Prove$(\mathsf{pp}, (\Phi, C), (\phi_\ell, r); \mathcal{T})$ *and* Verify$(\mathsf{pp}, (\Phi, C), \boldsymbol{\pi}; \mathcal{T})$*), which serves to establish a link between the proof and the specified $\mathcal{T}$. $\mathcal{T}$ will be utilized solely for generating randomness, i.e., as input to all random oracles, and does not influence the validity of the proof. This concept is inspired by the approach used by Groth et al. in their construction of ring signatures based on Σ-protocols [21]. We claim that such extension does not affect the security properties.*

4.2 Optimizing SMA for O(1) Exponentiations by the Verifier

In this section, we describe a technique in which the verifier's computation of multi-exponentiations is outsourced to the prover, allowing the verifier to perform a constant number of expensive operations, such as exponentiations and pairings. This can be achieved by exploiting the fact that the public parameters pp have a structure similar to that of KZG polynomial commitments [26], and utilizing the constant verification time of KZG. We rewrite the verification equation (14) as follows:

$$
\begin{aligned}
e(\pi, \bar{g}) = &e(C_s^{\delta_b} \cdot \underbrace{\prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i} + (\delta_{eq} \cdot t_i - \delta_b) \cdot s_i + \delta_{\phi,i} \cdot \phi_i}}_{\triangleq C_h}, \bar{C}_b) \cdot e(g_n^{\delta_o}, \bar{g}_1)^{-1} \\
&\underbrace{e(C, \bar{g}_n^{\delta_\phi} \cdot \bar{g}_{n-1}^{-\delta_{d2}} \cdot \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1} \cdot u_i})^{-1}}_{\triangleq \bar{C}_u} \cdot \underbrace{e(C_s, \prod_{i=1}^{n} \bar{g}_i^{\delta_{eq} \cdot t_i})^{-1}}_{\triangleq \bar{C}_t}
\end{aligned}
\tag{15}
$$

Note that $(C_h, \bar{C}_u, \bar{C}_t) = (g^{P_h(\beta)}, \bar{g}^{P_u(\beta)}, \bar{g}^{P_t(\beta)})$ can be viewed as deterministic KZG commitments of the following polynomials:

$$
\begin{aligned}
P_h[X] &= \sum_{i=1}^{n} (\delta_{o,i} + (\delta_{eq} \cdot t_i - \delta_b) \cdot s_i + \delta_{\phi,i} \cdot \phi_i) \cdot X^{n+1-i} \\
P_u[X] &= \delta_\phi \cdot X^n - \delta_{d2} \cdot X^{n-1} + \sum_{i=2}^{n} (-\delta_{d1} \cdot u_i) \cdot X^{n+1-i} \\
P_t[X] &= (\delta_{eq} \cdot t_i) \cdot X^n
\end{aligned}
\tag{16}
$$

The idea is to have the prover compute $(C_h, \bar{C}_u, \bar{C}_t)$ and convince the verifier that these are correctly computed KZG commitments. Specifically, this requires the prover to compute $(C_h, \bar{C}_u, \bar{C}_t)$ and include them in the proof, allowing the verifier to perform only one group exponentiation in $\mathbb{G}_1$ and one group exponentiation in $\mathbb{G}_2$. To ensure that $(C_h, \bar{C}_u, \bar{C}_t)$ are correctly computed KZG commitments derived from the aforementioned polynomials, the prover can open their evaluation at some random z, where z is derived from the hash of $(C_h, \bar{C}_u, \bar{C}_t)$ and $\Delta := (\Phi, \boldsymbol{s}, \boldsymbol{t}, \{u_i\}_{i\in[2,n]}, C, \bar{C}_b, C_s)$, i.e., other commitments and proof components. The prover processes as follows:

- Compute $P_h[X], P_u[X], P_t[X]$
- Compute $(C_h, \bar{C}_u, \bar{C}_t) = (g^{P_h(\beta)}, \bar{g}^{P_u(\beta)}, \bar{g}^{P_t(\beta)})$
- Sample $z \leftarrow\!\$\ H(\Delta, C_h, \bar{C}_u, \bar{C}_t) \in \mathbb{Z}_p$
- Evaluate $(p_h, p_u, p_t) = P_h[z], P_u[z], P_t[z]$
- Compute $(Q_h[X], Q_u[X], Q_t[X]) :=$

$$
\left(\frac{P_h[X] - p_h}{X - z}, \frac{P_u[X] - p_u}{X - z}, \frac{P_t[X] - p_t}{X - z}\right)
$$

- Sample $(\delta_h, \delta_u, \delta_t) \leftarrow\!\$\ H(\Delta, C_h, \bar{C}_u, \bar{C}_t, z)$
- Compute $\pi_{\mathsf{KZG}} = g^{\delta_h \cdot Q_h(\beta) + \delta_u \cdot Q_u(\beta) + \delta_t \cdot Q_t(\beta)} \in \mathbb{G}_1$
- Output $C_h, \bar{C}_u, \bar{C}_t, \pi_{\mathsf{KZG}}$

On input $(C_h, \bar{C}_u, \bar{C}_t, \pi_{\mathsf{KZG}})$, the verifier performs the following steps for verification:

- Compute $P_h[X], P_u[X], P_t[X]$
- Sample $z \leftarrow\!\$\ H(\Delta, C_h, \bar{C}_u, \bar{C}_t)$
- Evaluate $(p_h, p_u, p_t) = P_h[z], P_u[z], P_t[z]$
- Sample $(\delta_h, \delta_u, \delta_t) \leftarrow\!\$\ H(\Delta, C_h, \bar{C}_u, \bar{C}_t, z)$
- Check if

$$
e(C_h - g^{p_h}, \bar{g}^{\delta_h}) \cdot e(g^{\delta_u}, \bar{C}_u - \bar{g}^{p_u}) \cdot e(g^{\delta_t}, \bar{C}_t - \bar{g}^{p_t}) = e(\pi_{\mathsf{KZG}}, \bar{g}_1 \cdot \bar{g}^{-z})
\tag{17}
$$

Since all polynomials $P_h[X], P_u[X], P_t[X]$ are evaluated on a common input z, we can generate an aggregated proof π_{KZG} using random scalars $(\delta_h, \delta_u, \delta_t)$. The correctness of the above process is guaranteed by the correctness of KZG evaluation. If these commitments are malformed and commit to different polynomials $(Q_h[X], Q_u[X], Q_t[X]) \neq (P_h[X], P_u[X], P_t[X])$, the Schwartz-Zippel lemma ensures that $(Q_h(z), Q_u(z), Q_t(z)) = (p_h, p_u, p_t)$ with negligible probability. Detailed security analysis is given in Appendix A.4.

Efficiency. Now, the proof $\boldsymbol{\pi} := (\bar{C}_b, C_s, \pi, C_h, \bar{C}_u, \bar{C}_t, \pi_{\text{KZG}})$, i.e., 4 elements from $\mathbb{G}_1$ and 3 from $\mathbb{G}_2$. The prover's computation is dominated by $4n$ exponentiations in $\mathbb{G}_1$ (including n for each group element (C_h, π_{KZG}) and $2n$ to compute π) and $2n$ exponentiations in $\mathbb{G}_2$ (including n for each group element $(\bar{C}_u, \bar{C}_t)$). The verifier's computation now includes 5 exponentiations in $\mathbb{G}_1$, 4 exponentiations in $\mathbb{G}_2$, 9 pairings, and $O(n)$ field operations in $\mathbb{Z}_p$ to evaluate (p_h, p_u, p_t).

5 Construction of Scoped Linkable Ring Signatures

In this section, based on the constant size set membership argument SMA proposed in section 4, we provide a practical construction of a constant size linkable ring signature with efficient verification.

Our construction supports *scoped linkability* [23]. Let

- Let $\mathsf{SMA} := (\mathsf{PP\text{-}Gen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ be our set membership argument as presented in Section 4. The Prove and Verify algorithms are extended to accept an additional tag, as described in Remark 1.
- $\mathcal{H}_\phi, \mathcal{H}_L$: two collision-resistant cryptographic hash functions mapping $\{0,1\}^* \to \mathbb{Z}_p$.
- $\mathsf{cc}\Pi := (\mathsf{KG}, \mathsf{Prv}, \mathsf{Vrf}, \mathsf{VerC})$ be a ccSNARK for the relation $\mathcal{R}_{\text{kos}}$, which describes the knowledge of the private key sk_ℓ and its public key ϕ_ℓ. Let statement $x := (\mathsf{sc}, L)$, where L is the element used for linkability and sc is a scope string. Additionally, let $w = (u, v)$ be a witness, where the committed portion is $u := \phi_\ell$ and $v := \mathsf{sk}_\ell$. Furthermore, $\mathcal{M}$ denotes a public mapping. The relation $\mathcal{R}_{\text{kos}}$ is then defined as follows:

$$\mathcal{R}_{\text{kos}} = \left\{ ((\mathsf{sc}, L), (u := \phi_\ell, v := \mathsf{sk}_\ell)) : \begin{matrix} \phi_\ell = \mathcal{H}_\phi(\mathcal{M}(\mathsf{sk}_\ell)) \\ \wedge L = \mathcal{H}_L(\mathsf{sc}, \mathsf{sk}_\ell) \end{matrix} \right\}$$

- $\mathsf{lk}\Pi := (\mathsf{KG}, \mathsf{Prv}, \mathsf{Vrf})$ be a zkSNARK for relation $\mathcal{R}_{\mathsf{link}}$ as defined in section 3.4.

Let $(\mathsf{pk}', \mathsf{sk})$ be an arbitrary key pair within a PKI, satisfying $\mathsf{pk}' = \mathcal{M}(\mathsf{sk})$ for a public mapping $\mathcal{M}$. We require that it is computationally infeasible to derive sk from pk'. Since our SMA operates over $\mathbb{Z}_p$ and pk' is not necessarily an element of $\mathbb{Z}_p$, the signer employs a hash function $\mathcal{H}_\phi$ to hash pk' into $\phi \in \mathbb{Z}_p$. The public key pk of our LRS is defined as (pk', ϕ). We retain sk as the private key for our LRS. The ring $\mathsf{R} := (\phi_1, \ldots, \phi_m)$ is a public list of hashed public keys, and we require that each ϕ_i within the ring is derived using the same mapping $\mathcal{M}$.

For a high-level overview of the interactions among different components and the rationale behind their necessity in our linkable ring signature construction, please refer to the part of our ring signature framework in Section 2.

5.1 Scoped Linkable Ring Signature Framework

Consider a message m, a ring $R := (\phi_1, \ldots, \phi_m)$, and a scoped string sc. Let ϕ_ℓ represent the signer's public key, where ℓ is its position within the ring R.

In our Setup algorithm, given the security parameter λ and the upper bound of the ring size n, we first select a bilinear group $\mathcal{G}$. To ensure compatibility, we require SMA, $\mathsf{cc}\Pi$, and $\mathsf{lk}\Pi$ to operate within the same $\mathcal{G}$. We make this explicit by providing $\mathcal{G}$ as input to their respective setup algorithms. The public parameters pp of our LRS framework include the public parameters for SMA, $\mathsf{cc}\Pi$, and $\mathsf{lk}\Pi$.

$\mathsf{LRS.Setup}(1^\lambda, 1^n)$:
- Sample $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$
- Sample mapping $\mathcal{M}$ and $\mathcal{H}_\phi, \mathcal{H}_L$
- Compute $\mathsf{pp}_{\mathsf{sma}} := (\mathsf{crs}, \mathsf{ck}, \boldsymbol{H}) \leftarrow \mathsf{SMA.PP\text{-}Gen}\ (1^\lambda, 1^n, \mathcal{G})$
- Compute $(\mathsf{ck}_{\mathsf{cc}}, \mathsf{ek}_{\mathsf{cc}}, \mathsf{vk}_{\mathsf{cc}}) \leftarrow \mathsf{cc}\Pi.\mathsf{KG}(1^\lambda, \mathcal{G}, \mathcal{R}_{\text{kos}})$
- Compute $(\mathsf{ek}_{\mathsf{lk}}, \mathsf{vk}_{\mathsf{lk}}) \leftarrow \mathsf{lk}\Pi.\mathsf{KG}(1^\lambda, \mathcal{G}, \mathcal{R}_{\mathsf{link}}, \mathsf{ck}_{\mathsf{cc}}, \mathsf{ck})$
- Let $\mathsf{ek} := (\mathsf{pp}_{\mathsf{sma}}, \mathsf{ek}_{\mathsf{cc}}, \mathsf{ek}_{\mathsf{lk}})$, $\mathsf{vk} := (\mathsf{pp}_{\mathsf{sma}}, \mathsf{vk}_{\mathsf{cc}}, \mathsf{vk}_{\mathsf{lk}})$
- Output $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk})$

$\mathsf{LRS.KeyGen}(\mathsf{pp})$
- Sample $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$
- Compute $\mathsf{pk}' = \mathcal{M}(\mathsf{sk})$
- Compute $\phi \leftarrow \mathcal{H}_\phi(\mathsf{pk}') \in \mathbb{Z}_p$
- Output $(\mathsf{sk}, \mathsf{pk} := (\mathsf{pk}', \phi))$

$\mathsf{LRS.Sign}\ (\mathsf{pp}, \mathsf{sk}, \mathsf{m}, \mathsf{sc}, R := (\phi_1, \cdots, \phi_m))$
- Parse $\mathsf{pp.ek} := (\mathsf{pp}_{\mathsf{sma}}, \mathsf{ek}_{\mathsf{cc}}, \mathsf{ek}_{\mathsf{lk}})$
- Compute $(C_1, r) \leftarrow \mathsf{SMA.Com}(\mathsf{pp}_{\mathsf{sma}}, \phi_\ell)$
- Compute $\pi_{\mathsf{sma}} \leftarrow \mathsf{SMA.Prove}\ (\mathsf{pp}_{\mathsf{sma}}, (R, C_1), (\phi_\ell, r); \mathsf{m})$
- Compute $L \leftarrow \mathcal{H}_L(\mathsf{sc}, \mathsf{sk})$
- Let $\mathsf{cc}.x := (\mathsf{sc}, L)$, $\mathsf{cc}.w := (u := \phi_\ell, v := \mathsf{sk})$
- Compute $(C', \pi_{\mathsf{cc}}, o') \leftarrow \mathsf{cc}\Pi.\mathsf{Prv}\ (\mathsf{ek}_{\mathsf{cc}}, \mathsf{cc}.x, \mathsf{cc}.w)$
- Let $\boldsymbol{u}_1 := (\phi_\ell)$, $o_1 := r$
- Let $\mathsf{lk}.x := (C', (C_1))$, $\mathsf{lk}.w := ((\boldsymbol{u}_1), (o_1), o')$
- Compute $\pi_{\mathsf{lk}} \leftarrow \mathsf{lk}\Pi.\mathsf{Prv}(\mathsf{ek}_{\mathsf{lk}}, \mathsf{lk}.x, \mathsf{lk}.w)$
- Output $\sigma := (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L)$

$\mathsf{LRS.Verify}\ (\mathsf{pp}, \sigma, \mathsf{m}, \mathsf{sc}, R)$
- Parse $\mathsf{pp.vk} := (\mathsf{pp}_{\mathsf{sma}}, \mathsf{vk}_{\mathsf{cc}}, \mathsf{vk}_{\mathsf{lk}})$
- Parse $R := (\phi_1, \cdots, \phi_m)$
- Parse $\sigma := (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L)$
- Let $\mathsf{cc}.x := (\mathsf{sc}, L)$
- Let $\mathsf{lk}.x := (C', (C_1))$
- Output 1 if all following holds:
  - $\mathsf{SMA.Verify}(\mathsf{pp}_{\mathsf{sma}}, (R, C_1), \pi_{\mathsf{SMA}}) = 1$
  - $\mathsf{cc}\Pi.\mathsf{Vry}(\mathsf{vk}_{\mathsf{cc}}, \mathsf{cc}.x, C', \pi_{\mathsf{cc}}) = 1$
  - $\mathsf{lk}\Pi.\mathsf{Vrf}(\mathsf{vk}_{\mathsf{lk}}, \mathsf{lk}.x, \pi_{\mathsf{lk}}) = 1$
- Else output 0

$\mathsf{LRS.Link}\ (\mathsf{pp}, \sigma, \sigma', \mathsf{sc})$
- If $\sigma.L = \sigma'.L$, return 1
- Else output 0

Correctness. Our construction clearly satisfies the first point of the correctness definition in Section 3.5, namely that an honestly generated signature always passes verification. This follows from the intuition of our ring signature framework presented in

Section 2 and the correctness of its underlying components. Regarding the second criterion, which pertains to linkability, for any (σ, σ') honestly generated using the same sk for the same scope string sc, the Link algorithm always outputs 1. This is because both $\sigma.L$ and $\sigma'.L$ are deterministically generated from sk and sc using the cryptographic hash function $\mathcal{H}_L$.

Theorem 5.1. *In the ROM and the AGM, if the underlying* SMA, ccΠ *are zero-knowledge and simulation extractable,* lk *is zero-knowledge and (one-time) simulation extractable, the hash function $\mathcal{H}_\phi$ and the public mapping $\mathcal{M}$ are one-way and collision-resistant, and H_L is modeled as a random oracle, then the proposed LRS framework is scoped linkable, anonymous, and non-slanderable.*

We prove the Theorem 5.1 through the following lemma.

Lemma 5.2. *In the ROM and the AGM, our LRS framework is scoped linkable.*

Proof: Suppose there exists an adversary $\mathcal{A}$ that wins the linkability game defined in Definition B.2 with non-negligible advantage $\mathsf{Adv}^{\mathsf{LRS}}_{\mathsf{link}}(\mathcal{A})$. We construct a simulator $\mathcal{S}$ that leverages the success of $\mathcal{A}$ to solve the one-wayness and collision-resistant of $\mathcal{M}$.

$\mathcal{S}$ is given two challenge instances. The first instance corresponds to the one-wayness challenge, where $\mathcal{S}$ is given a output m_o and must output an x_o such that $m_o = \mathcal{M}(x_o)$. The second instance corresponds to the target collision-resistance challenge, where $\mathcal{S}$ is given an x_c and must output a different x'_c such that $\mathcal{M}(x_c) = m_c = \mathcal{M}(x'_c)$. The goal of $\mathcal{S}$ is to compute x_o or x'_c while simulating a signing system that is indistinguishable from the real-world game for $\mathcal{A}$. The simulator first generates the public parameters $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda, 1^n)$ and sends them to the adversary $\mathcal{A}$. The hash function $\mathcal{H}_L$ is modeled as a random oracle.

Oracle Simulation:

- KGO() : The adversary $\mathcal{A}$ queries KGO a total of q_j times. The simulator $\mathcal{S}$ randomly picks an index $q_c, q_o \in [1, \cdots, q_j]$. For the q_o-th query, $\mathcal{S}$ sets $pk_o = m_o$ and returns it to $\mathcal{A}$. For the q_c-th query, $\mathcal{S}$ sets $pk_c = m_c, sk_c = x_c$ and returns pk_c to $\mathcal{A}$. For all other queries, $\mathcal{S}$ runs $\mathsf{LRS.KeyGen(pp)}$ to generate $(\mathsf{sk}_i, \mathsf{pk}_i)$, returns pk_i, and records $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the list $\mathcal{L}$. Note that pk_o and pk_c are indistinguishable from other public keys in the adversary's view.
- $\mathsf{CO}(\mathsf{pk_i})$: Assume that $\mathcal{A}$ queries CO up to q_r times ($q_r < q$) to obtain the private key corresponding to a public key pk_i. If $\mathsf{pk}_i = pk_o$, $\mathcal{S}$ aborts. If $\mathsf{pk}_i = pk_c$, $\mathcal{S}$ returns sk_c. For other queries, $\mathcal{S}$ retrieves sk_i from $\mathcal{L}$ and returns it to $\mathcal{A}$.
- $\mathsf{SO}(\mathsf{sc_i}, R, \mathsf{pk_i}, \mathsf{m})$: For a query (R, pk_i, m), where $pk_i \in R$, $\mathcal{S}$ performs the following:
  - If $pk_i \neq pk_o$, $\mathcal{S}$ checks whether $(\mathsf{sk}_i, \mathsf{sc}_i)$ is already stored in the hash table $\mathcal{T}$. If it exists, it retrieves the corresponding tag L_i. If not, $\mathcal{S}$ randomly samples a tag L_i from $\mathbb{Z}_p$ and stores $(\{\mathsf{sk_i}, \mathsf{sc_i}\}, L_i, \mathsf{pk}_i)$ in $\mathcal{T}$. Then, $\mathcal{S}$ computes $(C_1, r) \leftarrow \mathsf{SMA.Com}(\mathsf{pp_{sma}}, \phi_\ell)$ and $\pi_{\mathsf{sma}} \leftarrow \mathsf{SMA.Prove}(\mathsf{pp_{sma}}, (R, C_1), (\phi_\ell, r); m)$.
  - If $pk_i = pk_o$, $\mathcal{S}$ randomly samples a tag L_i from $\mathbb{Z}_p$ and stores $(\{-, \mathsf{sc}_i\}, L_i, pk_o)$ in the hash table $\mathcal{T}$. S randomly samples a commitment C_1 from $\mathbb{G}_1$. Given $\mathsf{SMA}.x = (R, m, C_1)$, $\mathcal{S}$ invokes the simulator for the underlying SMA to generate a simulated proof: $\pi_{\mathsf{sma}} \leftarrow \mathsf{SMA.Sim}(\mathsf{pp}, \mathsf{SMA}.x)$. Next, $\mathcal{S}$ sets $\mathsf{cc}.x = (\mathsf{sc}_i, L_i)$ and invokes the simulator for the underlying ccΠ to generate a simulated proof $(C', \pi_{\mathsf{cc}}, o')$ by $(C', \pi_{\mathsf{cc}}, o') \leftarrow \mathsf{cc\Pi.Sim}(\mathsf{pp}, \mathsf{cc}.x, \tau)$. Then, $\mathcal{S}$ sets $\mathsf{lk}.x := (C', (C_1))$ and uses the simulator for the underlying lk scheme to generate a proof π_{lk}. Finally, the simulator $\mathcal{S}$ returns the signature $\sigma := (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L_i)$, and sends it to $\mathcal{A}$.
- $\mathcal{H}_L(\mathsf{sk_i}, \mathsf{sc_i})$: For any query (sk_i, sc_i), $\mathcal{S}$ first checks the hash table $\mathcal{T}$. If (sk_i, sc_i) exists, it returns the corresponding value. In addtion, if $\mathcal{M}(sk_i) = pk_o$, $\mathcal{S}$ detects that $sk_i = x_o$, then it outputs sk_i as the solution to the one-wayness challenge and retrieves L_i to $\mathcal{A}$ from $\mathcal{T}$. Otherwise, $\mathcal{S}$ randomly samples a tag L_i from $\mathbb{Z}_p$, adds $(\{\mathsf{sk_i}, \mathsf{sc_i}\}, L_i, -)$ to the hash table, and returns L_i.

Challenge Phase. $\mathcal{A}$ outputs a set of valid signatures $\{(\mathsf{sc}, \sigma_k, \mathsf{m_k}, R_k)\}_{k=1}^{q}$. According to the linkability game, at least one of the private keys used to generate these signatures must not have been queried to CO. Let sk^* denote such a private key, and let pk^* be its corresponding public key.

Analysis. For $\mathcal{A}$ to win the game, we can use the extractor of cc to extract the private keys $\{sk_i\}_{i=1}^{n}$ corresponding to these signatures due to simulation extractability property. it leads to the following two cases:

Case 1: sk^* corresponds to $pk_o = m_o$, and sk^* was not queried to CO. In this case, assume that the probability of success of $\mathcal{A}$ is $\Pr^1_{\mathcal{A}}$. By outputting sk^* by $\mathcal{A}$, $\mathcal{S}$ can return (sk^*, m_o) as the solution of the one-wayness challenge with the probability of success in this case is $\frac{q_j - q_r}{q_j} \cdot \frac{\Pr^1_{\mathcal{A}}}{q_j - \delta + 1}$, where δ denotes the number of distinct private keys queried to CO or $\mathcal{H}_L$. This probability of $\mathcal{S}$ is non-negligible, leading to a contradiction.

Case 2: $sk^* \neq sk'$ where sk' was queried, and $\mathcal{M}(sk^*) = \mathcal{M}(sk')$. In this case, $\mathcal{A}$ finds two distinct private keys sk^* and sk' that map to the same public key pk^*. assume that the probability of success of $\mathcal{A}$ is $\Pr^2_{\mathcal{A}}$. By outputting sk^* by $\mathcal{A}$, $\mathcal{S}$ can return (sk^*, m_c) as the solution of the collision-resistant challenge. The probability of success for $pk^* = m_c$ is $\frac{q_j - q_r}{q_j} \cdot \frac{\Pr^2_{\mathcal{A}}}{q_j}$. This probability of $\mathcal{S}$ is non-negligible, leading to a contradiction.

Lemma 5.3. *In the ROM and the AGM, our LRS framework is anonymous.*

Proof: Suppose there exists an adversary $\mathcal{A}$ that can win the anonymity game described in Definition B.1. To prove anonymity, we construct a simulator S that interacts with $\mathcal{A}$ and simulates the environment.

The simulator S first generates the public parameters $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda, 1^n)$, and sends pp to the adversary $\mathcal{A}$. The hash function $\mathcal{H}_L$ is modeled as a random oracle.

Oracle Simulation:

- KGO() : When $\mathcal{A}$ queries the oracle KGO, S runs $\mathsf{LRS.KeyGen(pp)}$ to generate $\mathsf{sk_i}, \mathsf{pk_i}$, and returns $\mathsf{pk_i}$ to $\mathcal{A}$. Then it records $(\mathsf{pk_i}, \mathsf{sk_i})$ in a list $\mathcal{L}$.
- $\mathsf{CO}(\mathsf{pk_i})$: For $\mathsf{pk_i} \in \mathcal{L}$, S returns the corresponding private key $\mathsf{sk_i}$.
- $\mathsf{SO}(\mathsf{sc_i}, R, \mathsf{pk_i}, \mathsf{m})$: S checks if $\mathsf{pk_i}, \mathsf{sc_i}$ exists in the hash table $\mathcal{T}$. If so, it retrieves the tag L_i; otherwise, it randomly samples a tag L_i from $\mathbb{Z}_p$ and records $(\{-, \mathsf{sc_i}\}, L_i, \mathsf{pk_i})$ in the hash

table $\mathcal{T}$. S then obtains $(C_1, r) \leftarrow \mathsf{SMA.Com}(\mathsf{pp}_{\mathsf{sma}}, \phi_\ell)$ and $\pi_{\mathsf{sma}} \leftarrow \mathsf{SMA.Prove}(\mathsf{pp}_{\mathsf{sma}}, (R, C_1), (\phi_\ell, r); \mathsf{m})$, and sets $\mathsf{cc}.x = (\mathsf{sc}_\mathsf{i}, L_i)$. It invokes the simulator for the underlying ccΠ to generate a simulated proof: $\mathsf{ccΠ.PP\text{-}Gen}(1^\lambda, \mathcal{G}, 1^n) \rightarrow (\mathsf{pp}, \tau)$, and $\mathsf{ccΠ.Sim}(\mathsf{pp}, \mathsf{cc}.x, \tau) \rightarrow (C', \pi_{\mathsf{cc}}, o')$. It then sets $\mathsf{lk}.x := (C', (C_1))$, and performs a similar simulation to generate π_{lk} using the simulator for the underlying lk scheme. Finally, S returns $\sigma := (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L_i)$ to $\mathcal{A}$.

- $\mathcal{H}_L(\mathsf{sk}_\mathsf{i}, \mathsf{sc}_\mathsf{i})$: For any query $\mathsf{sk}_\mathsf{i}, \mathsf{sc}_\mathsf{i}$ from $\mathcal{A}$, S checks if $pk_i = \mathcal{M}(\mathsf{sk}_\mathsf{i}), \mathsf{sc}_\mathsf{i}$ are stored in the hash table $\mathcal{T}$. If they exist, it returns the corresponding value; otherwise, it randomly samples a tag L_i from $\mathbb{Z}_p$, adds $(\{-, \mathsf{sc}_\mathsf{i}\}, L_i, pk_i)$ to the hash table, and returns L_i.

Challenge Phase: The adversary $\mathcal{A}$ chooses a ring $R = \{\mathsf{pk}_\mathsf{i}\}_{i\in[n]}$ containing public keys, an event identifier sc, a message m, and two indices i_0, i_1 such that $\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1} \in \mathcal{R}$, and submits $(R, \mathsf{sc}, \mathsf{m}, i_0, i_1)$ to S. The simulator S randomly samples a commitment C_1 from $\mathbb{G}_1$, and also randomly picks $b \in \{i_0, i_1\}$. Given $SMA.x = (R, \mathsf{m}, C_1)$, it invokes the simulator for the underlying SMA to generate a simulated proof π_{sma} by $\mathsf{SMA.PP\text{-}Gen}(1^\lambda, \mathcal{G}, 1^n) \rightarrow (\mathsf{pp})$, and $\mathsf{SMA.Sim}(\mathsf{pp}, \mathsf{SMA}.x) \rightarrow \pi_{\mathsf{sma}}$. Then, S randomly samples a tag L from $\mathbb{Z}_p$. It sets $\mathsf{cc}.x = (\mathsf{sc}, L)$, invokes the simulator for the underlying ccΠ to generate a simulated proof $(C', \pi_{\mathsf{cc}}, o')$, and generates π_{lk} using the simulator for the underlying lkΠ for $\mathsf{lk}.x := (C', (C_1))$, which acts as the same as in the random oracle. Finally, S returns $\sigma := (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L)$ to $\mathcal{A}$, and the adversary outputs a guess b'.

Analysis: Since the underlying commitment SMA.Com is a Pedersen commitment, which is known to possess the property of perfect hiding, it follows that the commitment generated by the simulator is indistinguishable from the real one. Furthermore, since SMA is zero-knowledge, the simulated proof π_{sma} is indistinguishable from the real proof. The same indistinguishability holds for (C', π_{cc}) and π_{lk} based on the zero-knowledge properties of the underlying ccΠ and lkΠ, respectively. Moreover, since the tag L is sampled uniformly at random, $\mathcal{A}$ gains no advantage in identifying the signer. Consequently, its success probability is $\Pr[b' = b] = \frac{1}{2}$, which establishes the anonymity of the framework.

Lemma 5.4. *In the ROM and the AGM, our LRS framework is non-slanderable.*

Proof: Suppose the adversary $\mathcal{A}$ can win the non-slanderability game described in Definition B.3 with non-negligible probability. Then, we can construct a simulator $\mathcal{S}$ that uses $\mathcal{A}$'s success to break the one-wayness of $\mathcal{H}_L$. Specifically, given a hash output h_t, the simulator $\mathcal{S}$ aims to find an input x such that $\mathsf{h}_\mathsf{t} = \mathcal{H}_L(x)$.

The simulator $\mathcal{S}$ first runs $\mathsf{pp} \leftarrow \mathsf{LRS.Setup}(1^\lambda, 1^n)$ and sends pp to the adversary $\mathcal{A}$.

Oracle Simulation:

- KGO() : For each query made by $\mathcal{A}$ to *KGO*, the simulator $\mathcal{S}$ runs $\mathsf{LRS.KeyGen}(\mathsf{pp})$ to generate a key pair $(\mathsf{sk}_\mathsf{i}, \mathsf{pk}_\mathsf{i})$. Then, $\mathcal{S}$ adds $(\mathsf{pk}_\mathsf{i}, \mathsf{sk}_\mathsf{i})$ to the list $\mathcal{L}$ and returns pk_i to $\mathcal{A}$.
- $\mathsf{CO}(\mathsf{pk}_\mathsf{i})$: For $\mathsf{pk}_\mathsf{i} \in \mathcal{L}$, S returns the corresponding private key sk_i.
- $\mathsf{SO}(\mathsf{sc}_\mathsf{i}, R, \mathsf{pk}_\mathsf{i}, \mathsf{m})$: For $\mathsf{pk}_\mathsf{i} \in \mathcal{L}$, $\mathcal{S}$ retrieves the corresponding sk_i. It then generates the signature by $\mathsf{LRS.Sign}(\mathsf{pp}, \mathsf{sk}_\mathsf{i}, \mathsf{m}, \mathsf{sc}_i, R) \rightarrow \sigma$. Additionally, $\mathcal{S}$ checks whether $\mathcal{H}_L(\mathsf{sk}_\mathsf{i}, \mathsf{sc}_\mathsf{i}) = \mathsf{h}_\mathsf{t}$. If so, $\mathcal{S}$ immediately outputs $(\mathsf{sk}_\mathsf{i}, \mathsf{sc}_\mathsf{i})$ as a solution to break the one-wayness of $\mathcal{H}_L$ and return σ to $\mathcal{A}$.

Challenge Phase. The adversary $\mathcal{A}$ outputs an scope identifier sc, a message m, a ring R and a public key $\mathsf{pk} \in R$. The simulator $\mathcal{S}$ sets $L = \mathsf{h}_\mathsf{t}$ and generates $(C_1, o_1) \leftarrow \mathsf{SMA.Com}(\mathsf{pp}_{\mathsf{sma}}, \phi_\ell)$ and $\pi_{\mathsf{sma}} \leftarrow \mathsf{SMA.Prove}(\mathsf{pp}_{\mathsf{sma}}, (R, C_1), (\phi_\ell, r); \mathsf{m})$. Then $\mathcal{S}$ sets $\mathsf{cc}.x = (\mathsf{sc}, L)$ and invokes the simulator of the ccΠ to generate $(C', \pi_{\mathsf{cc}}) \leftarrow \mathsf{ccΠ.Sim}(\mathsf{pp}, \mathsf{cc}.x, \tau)$. It then sets $\mathsf{lk}.x = (C', C_1)$ and invokes the simulator for lk to generate the linking proof π_{lk}. Finally, $\mathcal{S}$ returns the signature $\sigma = (C_1, C', \pi_{\mathsf{sma}}, \pi_{\mathsf{cc}}, \pi_{\mathsf{lk}}, L)$, to $\mathcal{A}$.

Analysis. If the adversary $\mathcal{A}$ outputs a valid signature σ', which includes a label L' such that $\mathsf{LRS.Verify}(\mathsf{sc}, \sigma', \mathsf{m}', R') = 1$ and pk was not queried via CO or SO, then the simulator can use the extractor of cc to extract a private key sk' such that $L' = \mathcal{H}_L(\mathsf{sk}', \mathsf{sc})$. Here, we use a one-time simulation-sound lk, noting that the adversary receives only one simulated signature during the challenge phase. In the simulation phase, all signatures provided to the adversary are honestly generated, not simulated.

Since $\mathsf{LRS.Link}(\mathsf{pp}, \sigma, \sigma', \mathsf{sc}) = 1$, it follows that $L' = L$. Hence, the simulator $\mathcal{S}$ has found $(\mathsf{sk}', \mathsf{sc})$ such that $\mathcal{H}_L(\mathsf{sk}', \mathsf{sc}) = \mathsf{h}_\mathsf{t}$, breaking the one-wayness of $\mathcal{H}_L$.

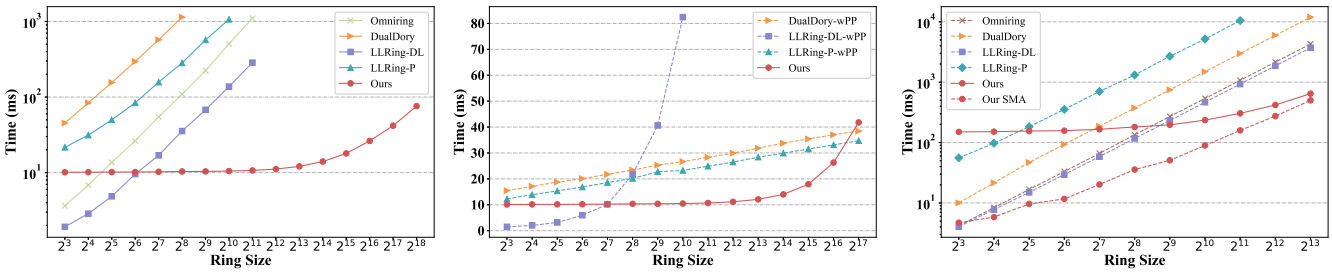
5.2 Instantiation and efficiency analysis

Instantiation. We instantiate ccΠ using ccGro16, the commit-carry variant of the Groth scheme [10, 20] achieving simulation extractability by employing techniques from [4], and lkΠ using $\mathsf{CP}_{\mathsf{link}}$ as described in Section 3.3 of [27]. For the following reasons, we are allowed to use ccGro16, the commit-carry variant of the Groth scheme with a non-updatable CRS [4, 10, 20], to instantiate ccΠ. The Setup algorithm first selects a bilinear group $\mathcal{G}$, followed by determining the mapping $\mathcal{M}$, and the hash functions $\mathcal{H}_\phi$ and $\mathcal{H}_L$. Note that once the above elements are fixed, $\mathcal{R}_{\mathsf{kos}}$ is uniquely determined. This means that different signers, even when selecting different rings, can share the common reference string $(\mathsf{ck}_{\mathsf{cc}}, \mathsf{ek}_{\mathsf{cc}}, \mathsf{vk}_{\mathsf{cc}})$ of ccΠ for relation $\mathcal{R}_{\mathsf{kos}}$ without regeneration. This allows us to adopt a cc-SNARK with non-updatable CRS, specifically ccGro16, rather than using updatable CRS schemes [13, 22], thereby achieving optimal proof size and verification efficiency.

Efficiency. We achieve a ring signature scheme with constant signature size and efficient verification. Specifically, an LRS signature consists of 10 elements from $\mathbb{G}_1$, 5 elements from $\mathbb{G}_2$, and 1 element from $\mathbb{Z}_p$.

	Prove/Sign	Verify	Comm.
SMA	$4n\,\mathbb{G}_1$ **Exps**, $2n\,\mathbb{G}_2$ **Exps**	$5\,\mathbb{G}_1$ **Exps**, $4\,\mathbb{G}_2$ **Exps**, $9\,\mathbb{P}$, $3n\,\mathbb{Z}_p$ **Muls**	$4\,\lvert\mathbb{G}_1\rvert$, $3\,\lvert\mathbb{G}_2\rvert$
ccΠ	$2w + d\,\mathbb{G}_1$ **Exps**, $w\,\mathbb{G}_2$ **Exps**	$1\,\mathbb{G}_2$ **Exps**, $4\,\mathbb{P}$	$3\,\lvert\mathbb{G}_1\rvert$, $2\,\lvert\mathbb{G}_2\rvert$
lkΠ	$12\,\mathbb{G}_1$ **Exps**	$2\,\mathbb{G}_2$ **Exps**, $4\,\mathbb{P}$	$2\,\lvert\mathbb{G}_1\rvert$
LRS	$4n + 2w + d + 12\,\mathbb{G}_1$ **Exps** $2n + w\,\mathbb{G}_2$ **Exps**	$5\,\mathbb{G}_1$ **Exps**, $7\,\mathbb{G}_2$ **Exps**, $17\,\mathbb{P}$, $3n\,\mathbb{Z}_p$ **Muls**	$10\,\lvert\mathbb{G}_1\rvert$, $5\,\lvert\mathbb{G}_2\rvert$, $1\,\lvert\mathbb{Z}_p\rvert$

Table 3: Efficiency of LRS and its components. n is the upper bound of the ring size, and m is the actual ring size. Let w and d denote the number of wires and multiplication gates in $\mathcal{C}_{\mathbf{kos}}$, respectively. Comm. is the abbreviation of communication complexity



(a) Verification for ad hoc rings. **(b) Verification for static rings.** **(c) Signing time**

Figure 1: Time cost comparison for various ring schemes with different ring sizes

This is achieved through the high efficiency of our building blocks. Due to the modular construction of LRS, its efficiency relies on the components SMA, ccΠ, and lkΠ. We instantiate the SMA as described in Section 4.2, ccΠ with ccGro16 [4, 10], and lkΠ with $\mathsf{CP}_{\mathsf{link}}$ from [27]. Notably, these components have constant communication complexity and require only a constant number of expensive operations, such as exponentiations and pairings, during verification. The efficiency parameters of each component and the overall efficiency statistics of LRS are summarized in Table 3.

In our LRS construction, lkΠ involves only one commitment value ϕ_ℓ, and two commitments C_1 and C'. Substituting these specific values into the efficiency parameters of $\mathsf{CP}_{\mathsf{Link}}$ in Section 3.4, the results shown in the table are derived. The cost of ccΠ depends on the parameters of the fixed arithmetic circuit $\mathcal{C}_{\mathsf{kos}}$ corresponding to the relation $\mathcal{R}_{\mathsf{kos}}$, i.e., the number of wires w and the number of multiplication gates d.

6 Evaluation

We evaluate the performance by running each test 100 times on Ubuntu 20.04.6 LTS with 16 GB RAM using a single CPU core. The code is publicly available[6].

Our LRS is implemented in Rust, utilizing the *BN*254 elliptic curve from the `ark-ec` library for the set membership argument (SMA). For the mapping $\mathcal{M}$ in signature generation, we use group exponentiation over the *BabyJubJub* curve[7], setting $\mathsf{pk}' = P^{\mathsf{sk}}$, where P is the generator of *BabyJubJub*. We instantiate $\mathcal{H}_\phi$ with the MiMC hash[8], both well-suited for SNARK circuits. By slightly modifying and integrating the implementations of `legogro16`[9] for the components ccΠ and lkΠ, we achieve our final LRS construction.

Signature Size. Our ring signature comprises 10 $\mathbb{G}_1$ elements, 5 $\mathbb{G}_2$ elements, and 1 $\mathbb{Z}_p$ element. Using the *BN*254 elliptic curve, where each $\mathbb{Z}_p$ element is 32 bytes, each $\mathbb{G}_1$ element is 33 bytes, each $\mathbb{G}_2$ element is 65 bytes, resulting in a constant-size ring signature of 687 bytes. This represents the state-of-the-art, as shown in Table 2.

In addition, we evaluate several state-of-the-art schemes, including LLRing [24], OmniRing [29], and DualDory [6], all of which exhibit efficient verification and are implemented using the *BN*254 curve. We observe that schemes like LLRing [24] (which includes two variants, LLRing-DL and LLRing-P) and DualDory [6] are mainly tailored to static-ring scenarios, enabling the verifier to precompute some expensive, ring-dependent operations (referred to as "offline computation"). As a result, the operations needed for each individual verification (termed "online computation") are substantially reduced. In our evaluation, we measure both cases: when the precomputation cost is counted as part of the verification time and when it is excluded.

Our test results are presented in Figure 1. We discuss the evaluation results as follows.

Verify Time. We measure the verification performance for both *ad hoc* and *static* rings, as shown in Figures 1b and 1a, respectively. Our verification time remains nearly constant for ring sizes below 2^{12}, primarily because only the SMA component in our scheme depends on the ring size. Specifically, SMA requires $\mathcal{O}(n)$ operations in $\mathbb{Z}_p$, which are several orders of magnitude faster than group exponentiations and pairings. The remaining verification steps involve a constant number of group exponentiations and pairings, which take around 10 ms.

In the ad hoc ring scenario, where pre-computation is not permitted, our approach achieves the best performance when the ring size larger than 64, and outperforms others for larger rings. This aligns with our scheme's advantages in large ad hoc rings. Even in the static ring setting, where other schemes are executed with pre-computation (marked with -wPP), our approach still achieves the best performance for ring sizes between 2^7 and 2^{16}. This demonstrates that our verification process scales efficiently with ring size, maintaining superior performance for both ad hoc and static rings.

Sign Time. The signature generation time is shown in Figure 1c. In our LRS, the total signing time consists of two parts: the ring-dependent set membership argument (SMA) and the ring-independent proof of private key knowledge. The cost of SMA grows linearly with the ring size, as demonstrated by the results in Figure 1c. The latter component introduces a fixed overhead of around 145.6 ms in our experiments. This fixed overhead explains the curvature observed in our LRS signing-time curve in Figure 1c. Once the SMA cost ($\mathcal{O}(n)$ group exponentiations) becomes dominant, the signing-time curve for our LRS begins to straighten, indicating the linear relationship between signing time and ring size. Although our scheme entails a fixed overhead, its lower linear coefficient ensures that when the ring size exceeds 512, our LRS offers the fastest signing time among the compared schemes.

[6]https://github.com/TuZhengzhou/lrs-v2.git
[7]https://github.com/iden3/circomlib/blob/master/circuits/babyjub.circom
[8]https://github.com/iden3/circomlib/blob/master/circuits/mimc.circom
[9]https://github.com/lovesh/legogro16

7 Conclusion

To explore more compact and verification-efficient linkable ring signatures, we propose a constant-size scoped linkable ring signature framework that features constant expensive operations during verification and strong compatibility and scalability for PKI cryptosystems. We design a novel constant-size Set Membership Argument (SMA) that requires only a constant number of group exponentiations and linear field multiplications in $\mathbb{Z}_p$ for the verifier. We prove the security of our framework under the ROM and the AGM and conduct benchmark tests to evaluate its performance.

The benchmark results show that when instantiated with public keys in a DL setting, our scheme achieves the shortest linkable ring signature size for ring sizes larger than 32. In ad-hoc ring scenarios, it demonstrates excellent performance in both signing and verification. In particular, compared to those SOTA ring signature schemes [6, 24, 29], it achieves the best verification performance for ring sizes larger than 64, and the best signing performance for ring sizes larger than 512, with its advantages becoming increasingly significant as the ring size grows.

Acknowledgments

We sincerely thank the shepherd and the anonymous reviewers for their constructive and invaluable feedback. This work was supported by the National Key R&D Program of China under Grant 2022YFB3102100, National Natural Science Foundation of China under Grant 62272131, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant 2022B1212010005, Shenzhen Science, Technology Major Project under Grant KJZD20230923114908017, The Hong Kong Polytechnic University under Grant P0046340, and RGC under Grant C1029-22GF and R1012-21.

References

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 2002. 1-out-of-n Signatures from a Variety of Keys. In *Proc. of ASIACRYPT*, Vol. 2501. 415–432.
[2] Anees Ahmed, Nojan Sheybani, Davi Moreno, Nges Brian Njungle, Tengkai Gong, Michel A. Kinsy, and Farinaz Koushanfar. 2024. AMAZE: Accelerated MiMC Hardware Architecture for Zero-Knowledge Applications on the Edge. In *Proc. of ICCAD*. 194:1–194:9.
[3] Man Ho Au, Willy Susilo, and Siu-Ming Yiu. 2006. Event-Oriented k-Times Revocable-iff-Linked Group Signatures. In *Proc. of ACISP*, Vol. 4058. 223–234.
[4] Karim Baghery, Zaira Pindado, and Carla Ràfols. 2020. Simulation Extractable Versions of Groth's zk-SNARK Revisited. In *Proc. of CANS*, Vol. 12579. 453–461.
[5] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *Proc. of IEEE S&P*. 459–474.
[6] Jonathan Bootle, Kaoutar Elkhiyaoui, Julia Hesse, and Yacov Manevich. 2022. DualDory: Logarithmic-Verifier Linkable Ring Signatures Through Preprocessing. In *Proc. of ESORICS*, Vol. 13555. 427–446.
[7] Priyanka Bose, Dipanjan Das, and Chandrasekharan Pandu Rangan. 2015. Constant size ring signature without random oracle. In *Proc. of ACISP*. 230–247.
[8] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *Proc. of IEEE S&P*. 315–334.
[9] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. 2021. Proofs for inner pairing products and applications. In *Proc. of ASIACRYPT*. 65–97.
[10] Matteo Campanelli, Dario Fiore, and Anaïs Querol. 2019. LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. In *Proc. of CCS*. 2075–2092.
[11] David Chaum and Eugène van Heyst. 1991. Group Signatures. In *Proc. of EUROCRYPT*, Vol. 547. 257–265.
[12] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. 2023. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In *Proc. of EUROCRYPT*, Vol. 14005. 499–530.
[13] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. 2020. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Proc. of EUROCRYPT*. 738–768.
[14] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. 2004. Anonymous Identification in Ad Hoc Groups. In *Proc. of EUROCRYPT*, Vol. 3027. 609–626.
[15] Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. 2016. Hash First, Argue Later: Adaptive Verifiable Computations on Outsourced Data. In *Proc. of CCS*. 1304–1316.
[16] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. 2018. The algebraic group model and its applications. In *Proc. of CRYPTO*. 33–62.
[17] Eiichiro Fujisaki and Koutarou Suzuki. 2007. Traceable ring signature. In *Proc. of PKC*. 181–200.
[18] Brandon Goodell, Sarang Noether, and Arthur Blue. 2019. Concise linkable ring signatures and forgery against adversarial keys. *Cryptology ePrint Archive* (2019).
[19] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. 2020. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proc. of CCS*. 2007–2023.
[20] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *Proc. of EUROCRYPT*. 305–326.
[21] Jens Groth and Markulf Kohlweiss. 2015. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Proc. of EUROCRYPT*. 253–280.
[22] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. 2018. Updatable and universal common reference strings with applications to zk-SNARKs. In *Proc. of CRYPTO*. 698–728.
[23] Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. 2022. Logarithmic-size (linkable) threshold ring signatures in the plain model. In *Proc. of PKC*. 437–467.
[24] Xiangyu Hui and Sid Chi-Kin Chau. 2024. tt LLRing: Logarithmic Linkable Ring Signatures with Transparent Setup. In *Proc. of ESORICS*, Vol. 14984. 299–319.
[25] Don Johnson, Alfred Menezes, and Scott Vanstone. 2001. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security* 1 (2001), 36–63.
[26] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. 2010. Constant-size commitments to polynomials and their applications. In *Proc. of ASIACRYPT*. 177–194.
[27] Eike Kiltz and Hoeteck Wee. 2015. Quasi-Adaptive NIZK for Linear Subspaces Revisited. In *Proc. of EUROCRYPT*, Vol. 9057. 101–128.
[28] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. 2017. A Traceability Analysis of Monero's Blockchain. In *Proc. of ESORICS*, Vol. 10493. 153–173.
[29] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: Scaling Private Payments Without Trusted Setup. In *Proc. of CCS*. 31–48.
[30] Benoît Libert. 2024. Vector Commitments with Proofs of Smallness: Short Range Proofs and More. In *Proc. of PKC*, Vol. 14602. 36–67.
[31] Benoit Libert and Moti Yung. 2010. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *Proc. of TCC*. 499–517.
[32] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. 2004. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In *Proc. of ACISP*, Vol. 3108. 325–335.
[33] Joseph K Liu and Duncan S Wong. 2005. Linkable ring signatures: Security models and new schemes. In *Proc. of ICCSA*. 614–623.
[34] Silvio Micali. 2000. Computationally sound proofs. *SIAM J. Comput.* 30, 4 (2000), 1253–1298.
[35] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. 2018. An Empirical Analysis of Traceability in the Monero Blockchain. *Proc. Priv. Enhancing Technol.* 2018, 3 (2018), 143–163.
[36] Lan Nguyen. 2005. Accumulators from Bilinear Pairings and Applications to ID-based Ring Signatures and Group Membership Revocation. *IACR Cryptol. ePrint Arch.* (2005), 123. http://eprint.iacr.org/2005/123
[37] Shen Noether. 2015. Ring SIgnature Confidential Transactions for Monero. *IACR Cryptol. ePrint Arch.* (2015), 1098.
[38] Torben Pryds Pedersen. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO*. 129–140.
[39] Ronald L. Rivest, Adi Shamir, and Yael Tauman. 2001. How to Leak a Secret. In *Proc. of ASIACRYPT*, Vol. 2248. 552–565.
[40] Yuxi Xue, Xingye Lu, Man Ho Au, and Chengru Zhang. 2024. Efficient linkable ring signatures: new framework and post-quantum instantiations. In *Proc. of ESORICS*. 435–456.
[41] Tsz Hon Yuen, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. 2021. DualRing: Generic Construction of Ring Signatures with Efficient Instantiations. In *Proc. of CRYPTO*, Vol. 12825. 251–281.
[42] Tsz Hon Yuen, Shifeng Sun, Joseph K. Liu, Man Ho Au, Muhammed F. Esgin, Qingzhao Zhang, and Dawu Gu. 2020. RingCT 3.0 for Blockchain Confidential Transaction: Shorter Size and Stronger Security. In *Proc. of FC*, Vol. 12059. 464–483.

A Security Proof of Our SMA

A.1 Correctness

Assume that $C = g^r \cdot g_1^{\phi_\ell}$ for $\phi_\ell \in \Phi := (\phi_1, \phi_2, \ldots, \phi_m)$ with the opening information $r \in \mathbb{Z}_p$, and $\boldsymbol{b} \in \{0,1\}^m$ is a vector that is 1 at index $\ell \in [1,m]$ and 0 elsewhere, $\bar{C}_b = \bar{g}^{r_b} \cdot \prod_{i=1}^m \bar{g}_i^{b_i}$ and $C_s = g^{r_s} \cdot \prod_{i=1}^m g_{n+1-i}^{s_i \cdot b_i}$ are correctly generated.

Equation (8) is satisfied by dividing the following two equalities:

$$
\begin{aligned}
e(\prod_{i=1}^n g_{n+1-i}^{t_i \cdot s_i}, \bar{C}_b) &= \prod_{i=1}^n e(g_{n+1-i}, \bar{C}_b)^{t_i \cdot s_i} \\
&= \prod_{i=1}^n (e(g_n, \bar{g}_1)^{b_i} \cdot e(g_{n+1-i}^{r_b} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j}, \bar{g}))^{t_i \cdot s_i} \\
&= e(g_n, \bar{g}_1)^{\sum_{i=1}^n b_i \cdot t_i \cdot s_i} \cdot e(\prod_{i=1}^n (g_{n+1-i}^{r_b} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j})^{t_i \cdot s_i}, \bar{g})
\end{aligned}
$$

and

$$
\begin{aligned}
e(C_s, \prod_{i=1}^n \bar{g}_i^{t_i}) &= \prod_{i=1}^n e(g^{r_s} \cdot \prod_{j=1}^m g_{n+1-j}^{s_j \cdot b_j}, \bar{g}_i)^{t_i} \\
&= \prod_{i=1}^n e(g_i^{r_s} \cdot \prod_{j=1}^m g_{n+1-j+i}^{s_j \cdot b_j}, \bar{g})^{t_i} \\
&= e(g_n, \bar{g}_1)^{\sum_{i=1}^n b_i \cdot t_i \cdot s_i} \cdot e(\prod_{i=1}^n (g_i^{r_s} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1+i-j}^{s_j \cdot b_j})^{t_i}, \bar{g})
\end{aligned}
$$

As for equation (9), we have

$$
\begin{aligned}
e(C_s \cdot \prod_{j=1}^n g_{n+1-j}^{-s_j}, \bar{C}_b) &= e(g^{r_s} \cdot \prod_{j=1}^n g_{n+1-j}^{s_j \cdot (b_j - 1)}, \bar{g}^{r_b} \cdot \prod_{i=1}^n \bar{g}_i^{b_i}) \\
&= e(g_n, \bar{g}_1)^{\sum_{i=1}^n s_j \cdot b_j \cdot (b_j - 1)} \cdot e(g^{r_b r_s} \cdot (\prod_{j=1}^n g_{n+1-j}^{r_b s_j (b_j - 1)}), \bar{g}) \\
&\cdot e(\prod_{i=1}^n (g_i^{b_i r_s} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-j+i}^{b_i s_j (b_j - 1)}), \bar{g}) \\
&= e(g^{r_b r_s} \cdot (\prod_{j=1}^n g_{n+1-j}^{r_b s_j (b_j - 1)}) \cdot \prod_{i=1}^n (g_i^{b_i r_s} \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-j+i}^{b_i s_j (b_j - 1)}), \bar{g}) \\
&= e(\pi_b, \bar{g})
\end{aligned}
$$

Equation (10) is satisfied by $\pi_o = \prod_{i=1}^m (g_{n+1-i}^r \cdot \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j})$ since:

$$
\begin{aligned}
e(\prod_{i=1}^m g_{n+1-i}, \bar{C}_b) &= e(\prod_{i=1}^m g_{n+1-i}, \bar{g}^{r_b} \cdot \prod_{j=1}^n \bar{g}_j^{b_j}) \\
&= e(g_n, \bar{g}_1)^{\sum_{i=1}^n b_i} \cdot e(\prod_{i=1}^m g_{n+1-i}^{r_b}, \bar{g}) \cdot e(\prod_{i=1}^m \prod_{j \in [n]\setminus\{i\}} g_{n+1-i+j}^{b_j}, \bar{g}) \\
&= e(g_n, \bar{g}_1) \cdot e(\pi_o, \bar{g})
\end{aligned}
$$

It is easy to see that equation (11) satisfies since:

$$
e(C, \bar{g}_{n-1}) = e(g^r \cdot g_1^{\phi_\ell}, \bar{g}_{n-1}) = e(g_{n-1}^r g_n^{\phi_\ell}, \bar{g})
$$

Equation (12) is satisfied by $\pi_{d1} = \prod_{i=2}^n (g_{n+1-i}^r \cdot g_{n+2-i}^{\phi_\ell})^{u_i}$ since:

$$
\begin{aligned}
e(C, \prod_{i=2}^n \bar{g}_{n+1-i}^{u_i}) &= \prod_{i=2}^n e(g^r \cdot g_1^{\phi_\ell}, \bar{g}_{n+1-i})^{u_i} \\
&= e(\prod_{i=2}^n (g_{n+1-i}^r \cdot g_{n+2-i}^{\phi_\ell})^{u_i}, \bar{g}) \\
&= e(\pi_{d1}, \bar{g})
\end{aligned}
$$

And for equation (13), we have:

$$
\begin{aligned}
e(\prod_{i=1}^m g_{n+1-i}^{\phi_i}, \bar{C}_b) &= \prod_{i=1}^m e(g_{n+1-i}^{\phi_i}, \bar{g}^{r_b} \cdot \prod_{j=1}^n \bar{g}_j^{b_j}) \\
&= \prod_{i=1}^m e(g_{n+1-i}^{\phi_i \cdot r_b} \cdot \prod_{j=1}^n g_{n+1-i+j}^{\phi_i \cdot b_j}, \bar{g}) \\
&= e(g_n, \bar{g}_1)^{\sum_{i=1}^m \phi_i \cdot b_i} \cdot e(\prod_{i=1}^m (g_{n+1-i}^{\phi_i \cdot r_b} \cdot \prod_{j=1, j \neq i}^n g_{n+1-i+j}^{\phi_i \cdot b_j}), \bar{g})
\end{aligned}
$$

Recall that $\phi_\ell = \sum_{i=1}^m \phi_i \cdot b_i$ and $e(C, \bar{g}_n) = e(g_n^r, \bar{g}) \cdot e(g_n, \bar{g}_1)^{\phi_\ell}$, which implies that dividing the two equations yields $e(\pi_\phi, \bar{g}) = e(\prod_{i=1}^m g_{n+1-i}^{\phi_i}, \bar{C}_b) / e(C, \bar{g}_n)$.

A.2 Proof of Theorem 4.1

Proof: In the ROM, we describe a simulator $\mathcal{S}$ that perfectly simulates proofs by programming the random oracles and using the trapdoor $\beta \in \mathbb{Z}_p$. For a given commitment $C \in \mathbb{G}_1$ and a set $\Phi := (\phi_1, \phi_2, \ldots, \phi_m)$, $\mathcal{S}$ works as follows:

- Compute $\bar{C}_b = \bar{g}^{\theta_b} \in \mathbb{G}_2$ for $\theta_b \leftarrow\$ \mathbb{Z}_p$
- Sample $\boldsymbol{s} := (s_1, \ldots, s_n) \leftarrow\$ \mathbb{Z}_p^n$, $\boldsymbol{t} := (t_1, \ldots, t_n) \leftarrow\$ \mathbb{Z}_p^n$
- Sample $u_i \leftarrow\$ \mathbb{Z}_p$ for $i \in [2,n]$, $\boldsymbol{\delta} := (\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) \leftarrow\$ \mathbb{Z}_p^6$
- Set $\delta_{o,i} = \delta_o$ and $\delta_{\phi,i} = \delta_\phi$ for $i \in [m]$
- Set $\delta_{o,i} = \delta_{\phi,i} = \phi_i = 0$ for $i \in [m+1, n]$
- Compute

$$
\mu_1 = \frac{\delta_\phi \cdot \beta^n - \delta_{d2} \cdot \beta^{n-1} - \sum_{i=2}^n \delta_{d1} \cdot u_i \cdot \beta^{n+1-i}}{\theta_b \delta_b + \sum_{i=1}^n (\delta_{eq} t_i) \cdot \beta^{n+1-i}}
$$

$$
\mu_2 = \frac{\delta_o \cdot \beta^n}{\theta_b \delta_b + \sum_{i=1}^n (\delta_{eq} t_i) \cdot \beta^{n+1-i}}
$$

- Compute $C_s = C^{\mu_1} \cdot g_1^{\mu_2} \cdot g^\gamma$ for $\gamma \leftarrow\$ \mathbb{Z}_p$
- Compute

$$
\pi = \left(g^{\theta_b \delta_b} \cdot \prod_{i=1}^n g_i^{-\delta_{eq} \cdot t_i} \right)^\gamma \cdot \left(\prod_{i=1}^n g_{n+1-i}^{\delta_{o,i} + (\delta_{eq} \cdot t_i - \delta_b) \cdot s_i + \delta_{\phi,i} \cdot \phi_i} \right)^{\theta_b}
$$

- Program the random oracles to have:
  - $\boldsymbol{s} := (s_1, \ldots, s_n) = H_s(C, \bar{C}_b)$
  - $\boldsymbol{t} := (t_1, \ldots, t_n) = H_t(\boldsymbol{s}, C, \bar{C}_b, C_s)$
  - $u_i = H_u(i, [2,n], C, \bar{C}_b, C_s) \in \mathbb{Z}_p$ for $i \in [2,n]$
  - $\boldsymbol{\delta} := (\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) = H_{\text{agg}}(C, \bar{C}_b, C_s)$

Note that the denominator of μ_1 and μ_2 is uniformly distributed over $\mathbb{Z}_p$ and non-zero with probability $1 - 1/p$, $\bar{C}_b$ and C_s are

uniformly distributed over $\mathbb{G}_2$ and $\mathbb{G}_1$, and π does not use the g_{n+1} term.

This provide a valid proof $\boldsymbol{\pi} := (\bar{C}_b, C_s, \pi)$. Specifically, π satisfies the verification equation (14) since we have[10]

$$\bar{g}_n^{\delta_\phi} \cdot \bar{g}_{n-1}^{-\delta_{d2}} \cdot \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1}\cdot u_i} = (\bar{C}_b^{\delta_b} \cdot \prod_{i=1}^{n} \bar{g}_i^{\delta_{eq}\cdot t_i})^{\mu_1}$$

$$\bar{g}_n^{\delta_o} = (\bar{C}_b^{\delta_b} \cdot \prod_{i=1}^{n} \bar{g}_i^{\delta_{eq}\cdot t_i})^{\mu_2}$$

which implies

$$\begin{aligned}
&\frac{e(C_s^{\delta_b} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i}+(\delta_{eq}\cdot t_i-\delta_b)\cdot s_i+\delta_{\phi,i}\cdot\phi_i}, \bar{C}_b)}{e(C, \bar{g}_n^{\delta_\phi}\bar{g}_{n-1}^{-\delta_{d2}} \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1}\cdot u_i}) \cdot e(C_s, \prod_{i=1}^{n} \bar{g}_i^{\delta_{eq}\cdot t_i},) \cdot e(g_n^{\delta_o}, \bar{g}_1)} \\
&= \frac{e(C_s, \bar{C}_b^{\delta_b} \cdot \prod_{i=1}^{n} \bar{g}_i^{-\delta_{eq}\cdot t_i}) \cdot e(\prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i}+(\delta_{eq}\cdot t_i-\delta_b)\cdot s_i+\delta_{\phi,i}\cdot\phi_i}, \bar{C}_b)}{e(C, \bar{g}_n^{\delta_\phi}\bar{g}_{n-1}^{-\delta_{d2}} \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1}\cdot u_i}) \cdot e(g_n^{\delta_o}, \bar{g}_1)} \\
&= \frac{e(C^{\mu_1} g_1^{\mu_2} g^{\gamma}, \bar{C}_b^{\delta_b} \prod_{i=1}^{n} \bar{g}_i^{-\delta_{eq}\cdot t_i}) \cdot e(\prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i}+(\delta_{eq}\cdot t_i-\delta_b)\cdot s_i+\delta_{\phi,i}\cdot\phi_i}, \bar{C}_b)}{e(C, \bar{g}_n^{\delta_\phi} \cdot \bar{g}_{n-1}^{-\delta_{d2}} \cdot \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1}\cdot u_i}) \cdot e(g_n^{\delta_o}, \bar{g}_1)} \\
&= \frac{e(C, (\bar{C}_b^{\delta_b} \cdot \prod_{i=1}^{n} \bar{g}_i^{-\delta_{eq}\cdot t_i})^{\mu_1})}{e(C, \bar{g}_n^{\delta_\phi} \cdot \bar{g}_{n-1}^{-\delta_{d2}} \cdot \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1}\cdot u_i})} \cdot \frac{e(g_1, (\bar{C}_b^{\delta_b} \cdot \prod_{i=1}^{n} \bar{g}_i^{-\delta_{eq}\cdot t_i})^{\mu_2})}{e(g_n^{\delta_o}, \bar{g}_1)} \\
&\quad \cdot e((g^{\theta_b\cdot\delta_b} \cdot \prod_{i=1}^{n} g_i^{-\delta_{eq}\cdot t_i})^{\gamma}, \bar{g}) \cdot e((\prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i}+(\delta_{eq}\cdot t_i-\delta_b)\cdot s_i+\delta_{\phi,i}\cdot\phi_i})^{\theta_b}, \bar{g}) \\
&= e(\pi, \bar{g})
\end{aligned}$$

A.3 Proof of Theorem 4.2

In the ROM and AGM models, assume the hardness of the $(2n, n)$-dlog assumption. For a statement $x := (C, \Phi := (\phi_1, \ldots, \phi_m))$ and a proof $\boldsymbol{\pi} := (\bar{C}_b, C_s, \pi)$ generated by any adversary $\mathcal{A}$, we show that there exists an extractor that can produce a valid witness.

Queries. At each random oracle query, $\mathcal{B}$ returns a random element in the appropriate range. When $\mathcal{A}$ queries a hash value $H_{\mathrm{agg}}(C, \bar{C}_b, C_s)$, $\mathcal{B}$ makes the corresponding queries

$$\begin{aligned}
\boldsymbol{s} &= H_s(C, \bar{C}_b), \\
\boldsymbol{t} &= H_t(\boldsymbol{s}, C, \bar{C}_b, C_s), \\
\{u_i &= H_u(i, [2, n], C, \bar{C}_b, C_s))\}_{i=2}^{n}
\end{aligned}$$

on its own before returning a tuple $(\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi)$. At the first query involving a group element, $\mathcal{A}$ provides a representation of that group element as a linear combination of all group elements observed so far in the same group.

At any time, $\mathcal{A}$ can choose a commitment C and ask for a simulated proof that C is a commitment to some public key in ring $\Phi := (\phi_1, \phi_2, \ldots, \phi_m)$ for some $m \le n$ of its choice. Since $\mathcal{A}$ is algebraic, it provides a representation of C with respect to $\{g_i\}_{i\in[0,2n]\setminus n+1}$ and the commitments C_s appearing in earlier simulated proofs. However, the simulator used by $\mathcal{B}$ is itself algebraic and always simulates proofs by computing C_s as a linear combination of $\{g_i\}_{i\in[0,2n]\setminus n+1}$ for coefficients of its choice. Hence, for any C chosen by $\mathcal{A}$, $\mathcal{B}$ can always compute a representation $\{a_i\}_{i\in[0,2n]\setminus n+1}$ such that

$$C = g^{a_0} \prod_{i\in[1,2n]\setminus n+1} g_i^{a_i}.$$

We assume without loss of generality that either $a_1 \notin \Phi$ or $(a_2, \ldots, a_n, a_{n+2}, \ldots, a_{2n}) \neq 0$, since otherwise $\mathcal{B}$ can generate a real proof using (a_1, a_0). Then, $\mathcal{B}$ simulates a proof as follows without using g_{n+1}:

(1) Choose random vectors $\boldsymbol{\delta} = (\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) \leftarrow\$ \mathbb{Z}_p^6$, $\boldsymbol{s} = (s_1, \ldots, s_n) \leftarrow\$ \mathbb{Z}_p^n$, $\boldsymbol{t} = (t_1, \ldots, t_n) \leftarrow\$ \mathbb{Z}_p^n$, $(u_2, \ldots, u_n) \leftarrow\$ \mathbb{Z}_p^{n-1}$.

(2) Choose random $c_0, d_0 \leftarrow\$ \mathbb{Z}_p$ and compute simulated commitments

$$\bar{C}_b = \prod_{i=0}^{n} \bar{g}_i^{c_i}, \quad C_s = \prod_{i=0,i\neq n+1}^{2n} g_i^{d_i}$$

We provide the definition of $(c_1, \ldots, c_n)$ and $(d_1, \ldots, d_n, d_{n+2}, \ldots, d_{2n})$ later.

(3) Define the polynomials

$$\begin{aligned}
Q_{eq}[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{n} t_i \cdot s_i \cdot X^{n+1-i}\right) \\
&\quad - \left(\sum_{i=1}^{n} t_i \cdot X^i\right) \cdot \left(\sum_{i=0,i\neq n+1}^{2n} d_i \cdot X^i\right) = \sum_{i=0}^{3n} \omega_i \cdot X^i \\
Q_b[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=0,i\neq n+1}^{2n} d_i \cdot X^i - \sum_{i=1}^{n} s_i \cdot X^{n+1-i}\right) \\
&= \sum_{i=0}^{3n} \varphi_i \cdot X^i \\
Q_o[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{m} X^{n+1-i}\right) - X^{n+1} = \sum_{i=0}^{n+m} \rho_i \cdot X^i \\
Q_{d2}[X] &= \left(\sum_{i=0,i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot X^{n-1} = \sum_{i=n-1}^{3n-1} \mu_i \cdot X^i \\
Q_{d1}[X] &= \left(\sum_{i=0,i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot \left(\sum_{i=2}^{n} u_i \cdot X^{n+1-i}\right) = \sum_{i=0}^{3n} \gamma_i \cdot X^i \\
Q_\phi[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{m} \phi_i \cdot X^{n+1-i}\right) \\
&\quad - \left(\sum_{i=0,i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot X^n = \sum_{i=0}^{3n-1} \eta_i \cdot X^i \\
Q_{\mathrm{agg}}[X] &= \delta_{eq} \cdot Q_{eq}[X] + \delta_b \cdot Q_b[X] + \delta_o \cdot Q_o[X] \\
&\quad + \delta_{d2} \cdot Q_{d2}[X] + \delta_{d1} \cdot Q_{d1}[X] + \delta_\phi \cdot Q_\phi[X] \\
&= \sum_{i=0}^{3n} \theta_i \cdot X^i
\end{aligned}$$

[10] Here, different colors are used to enhance clarity and understanding.

Next, we distinguish between the following two cases: $a_{2n} \neq 0$ and $a_{2n} = 0$.

Case (1): $a_{2n} \neq 0$. Define $c_i = 0, \forall i \in [2, n]$, $d_n = s_1$. Then we obtain the coefficients of above polynomials as follows. We use corresponding color to mark the consequence of such definition in following coefficients.

Their degree-(3n) coefficients are

$$\begin{aligned}
\omega_{3n} &= t_n \cdot d_{2n} \\
\varphi_{3n} &= c_n \cdot d_{2n} = 0 \\
\rho_{3n} &= 0 \\
\mu_{3n} &= 0 \\
\gamma_{3n} &= 0 \\
\eta_{3n} &= -a_{2n}
\end{aligned}$$

Their degree-(2n+k) coefficients are $(2 \leq k \leq n-1)$

$$\begin{aligned}
\omega_{2n+k} &= \sum_{i=k}^{n} t_i \cdot d_{2n+k-i} \\
\varphi_{2n+k} &= \sum_{i=k}^{n} c_i \cdot d_{2n+k-i} = 0 \\
\rho_{2n+k} &= 0 \\
\mu_{2n+k} &= a_{n+k+1} \\
\gamma_{2n+k} &= \sum_{i=2}^{n+1-k} u_i \cdot a_{n+k-1+i} \\
\eta_{2n+k} &= -a_{n+k}
\end{aligned}$$

Their degree-(2n+1) coefficients are

$$\begin{aligned}
\omega_{2n+1} &= \sum_{i=1}^{n-1} t_i \cdot d_{2n+1-i} \\
\varphi_{2n+1} &= \sum_{i=1}^{n-1} c_i \cdot d_{2n+1-i} = c_1 \cdot d_{2n} \\
\rho_{2n+1} &= 0 \\
\mu_{2n+1} &= a_{n+2} \\
\gamma_{2n+1} &= \sum_{i=2}^{n} u_i \cdot a_{n+i} \\
\eta_{2n+1} &= 0
\end{aligned}$$

Their degree-(n+1) coefficients are

$$\begin{aligned}
\omega_{n+1} &= \sum_{i=1}^{n} t_i \cdot (s_i \cdot c_i - d_{n+1-i}) = t_1 \cdot s_1 \cdot c_1 - \sum_{i=1}^{n} t_i \cdot d_{n+1-i} \\
&= t_1 \cdot (s_1 \cdot c_1 - s_1) - \sum_{i=2}^{n} t_i \cdot d_{n+1-i} \\
\varphi_{n+1} &= \sum_{i=1}^{n} c_i \cdot (d_{n+1-i} - s_i) = c_1 \cdot (d_n - s_1) = 0 \\
\rho_{n+1} &= \sum_{i=1}^{m} c_i - 1 = c_1 - 1 \\
\mu_{n+1} &= a_2 \\
\gamma_{n+1} &= \sum_{i=2}^{n} a_i \cdot u_i \\
\eta_{n+1} &= \sum_{i=1}^{m} \phi_i \cdot c_i - a_1 = \phi_1 \cdot c_1 - a_1
\end{aligned}$$

And we make the following assignments in sequence:

(1) Let $\eta_{3n} = -a_{2n}$. Define $d_{2n} = -\frac{\delta_\phi \cdot \eta_{3n}}{\delta_{eq} \cdot t_n}$, which cancels the degree-(3n) term of $Q_{\text{agg}}[X]$. Note that $d_{2n} \neq 0$ w.h.p. since we now consider the case where $a_{2n} \neq 0$.
(2) For $k \in [2, n-1]$, define $d_{n+k} = -\frac{\delta_{eq} \cdot (\sum_{i=k}^{n-1} t_i \cdot d_{2n+k-i}) + \delta_{d2} \cdot \mu_{2n+k} + \delta_{d1} \cdot \gamma_{2n+k} + \delta_\phi \cdot \eta_{2n+k}}{\delta_{eq} \cdot t_n}$, which cancels the θ_{2n+k} term.
(3) Define $c_1 = -\frac{\delta_{eq} \cdot \omega_{2n+1} + \delta_{d2} \cdot \mu_{2n+1} + \delta_{d1} \cdot \gamma_{2n+1}}{\delta_{eq} \cdot d_{2n}}$, which cancels the θ_{2n+1} term. Note that $c_1 \neq 0$ w.h.p., since $a_{2n} \neq 0$ and γ_{2n+1} contains $u_n \cdot a_{2n}$.
(4) Compute an arbitrary vector $(d_1, \ldots, d_{n-1})$ satisfying $\delta_{eq} \cdot \omega_{n+1} + \delta_o \cdot \rho_{n+1} + \delta_{d2} \cdot \mu_{n+1} + \delta_{d1} \cdot \gamma_{n+1} + \delta_\phi \cdot \eta_{n+1} = 0$, which cancels the θ_{n+1} term.

Next, compute $\pi = \sum_{i=0, i \neq n+1}^{2n} g_i^{\theta_i}$ using $g, \{g_i\}_{i \in [2n] \setminus \{n+1\}}$ and return the simulated proof $\boldsymbol{\pi} = (\bar{C}_b, C_s, \pi)$.

Note that the simulated proof $\boldsymbol{\pi}$ satisfies the verification equation

$$\frac{e(C_s^{\delta_b} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{o,i} + (\delta_{eq} \cdot t_i - \delta_b) \cdot s_i + \delta_{\phi,i} \cdot \phi_i}, \bar{C}_b)}{e(C, \bar{g}_n^{\delta_\phi} \bar{g}_{n-1}^{-\delta_{d2}} \prod_{i=2}^{n} \bar{g}_{n+1-i}^{-\delta_{d1} \cdot u_i}) \cdot e(C_s, \prod_{i=1}^{n} \bar{g}_i^{\delta_{eq} \cdot t_i}) \cdot e(g_n^{\delta_o}, \bar{g}_1)} = e(\pi, \bar{g}), \tag{18}$$

and the proof $\boldsymbol{\pi}$ has the same distribution as a proof generated by the zero-knowledge simulator in the proof of Theorem 4.1. Indeed, π is uniquely determined by the commitments $(C, \bar{C}_b, C_s)$ and the $\mathbb{Z}_p$-elements $\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\delta}$ and $\{u_i\}_{i \in [2,n]}$ in (18). Also, while the committed vectors $\boldsymbol{c} \in \mathbb{Z}_p^n, \boldsymbol{d} \in \mathbb{Z}_p^{2n}$ are programmed in a special way, they are perfect hidden by the randomness c_0 and d_0 in $\bar{C}_b, C_s$).

Consequently, the simulation is perfect, unless a collision occurs when random oracles are programmed in one of the simulation queries. If Q_S (resp. Q_H) denotes the number of queries made by $\mathcal{A}$ to the simulator (resp. to random oracles), this happens with probability at most $(Q_S + Q_H) \cdot Q_H / p$.

Next we consider the case where $a_{2n} = 0$, but $a_1 \notin \Phi$ or $(a_2, \ldots, a_n, a_{n+2}, \ldots, a_{2n-1}) \neq 0$.

Case (2). $a_{2n} = 0$. Similarly as previous case, we define $c_i = 0, \forall i \in [2, n-1]$ (note that no $c_n = 0$) and $d_n = s_1$. Additionally, we define $d_1 = s_n$. Then we obtain the coefficients of above polynomials as follows. We use corresponding color to mark the results of such definition in following coefficients.

Their degree-(3n) coefficients are

$$\omega_{3n} = t_n \cdot d_{2n}, \qquad \varphi_{3n} = c_n \cdot d_{2n}$$
$$\rho_{3n} = 0, \qquad \mu_{3n} = 0$$
$$\gamma_{3n} = 0, \qquad \eta_{3n} = -a_{2n} = 0$$

Their degree-(2n+k) coefficients are $(2 \leq k \leq n-1)$

$$\omega_{2n+k} = \sum_{i=k}^{n} t_i \cdot d_{2n+k-i}$$
$$\varphi_{2n+k} = \sum_{i=k}^{n} c_i \cdot d_{2n+k-i}$$
$$\rho_{2n+k} = 0$$
$$\mu_{2n+k} = a_{n+k+1}$$
$$\gamma_{2n+k} = \sum_{i=2}^{n+1-k} u_i \cdot a_{n+k-1+i}$$
$$\eta_{2n+k} = -a_{n+k}$$

Their degree-(2n+1) coefficients are

$$\omega_{2n+1} = \sum_{i=1}^{n-1} t_i \cdot d_{2n+1-i}$$
$$\varphi_{2n+1} = \sum_{i=1}^{n-1} c_i \cdot d_{2n+1-i} = c_1 \cdot d_{2n}$$
$$\rho_{2n+1} = 0$$
$$\mu_{2n+1} = a_{n+2}$$
$$\gamma_{2n+1} = \sum_{i=2}^{n} u_i \cdot a_{n+i}$$
$$\eta_{2n+1} = 0$$

Their degree-(n+1) coefficients are

$$\omega_{n+1} = \sum_{i=1}^{n} t_i \cdot (s_i \cdot c_i - d_{n+1-i}) = t_1 \cdot s_1 \cdot c_1 - \sum_{i=1}^{n} t_i \cdot d_{n+1-i}$$
$$= t_1 \cdot (s_1 \cdot c_1 - s_1) - \sum_{i=2}^{n} t_i \cdot d_{n+1-i}$$
$$\varphi_{n+1} = \sum_{i=1}^{n} c_i \cdot (d_{n+1-i} - s_i) = c_1 \cdot (d_n - s_1) + c_n \cdot (d_1 - s_n) = 0$$
$$\rho_{n+1} = \sum_{i=1}^{m} c_i - 1 = c_1 - 1$$
$$\mu_{n+1} = a_2$$
$$\gamma_{n+1} = \sum_{i=2}^{n} a_i \cdot u_i$$
$$\eta_{n+1} = \sum_{i=1}^{m} \phi_i \cdot c_i - a_1 = \phi_1 \cdot c_1 - a_1$$

And we make the following assignments in sequence:

(1) Define $c_n = -t_n, d_{2n} \leftarrow\!\$\ \mathbb{Z}^+$. Note that this assignment cancels the θ_{3n} term.
(2) For $k \in [2, n-1]$, define $d_{n+k} = -\frac{\delta_{eq} \cdot (\sum_{i=k}^{n-1} t_i \cdot d_{2n+k-i}) + \delta_{d2} \cdot \mu_{2n+k} + \delta_{d1} \cdot \gamma_{2n+k} + \delta_{\phi} \cdot \eta_{2n+k}}{\delta_{eq} \cdot t_n}$, which cancels the θ_{2n+k} term.
(3) Define $c_1 = -\frac{\delta_{eq} \cdot \omega_{2n+1} + \delta_{d2} \cdot \mu_{2n+1} + \delta_{d1} \cdot \gamma_{2n+1}}{\delta_{eq} \cdot d_{2n}}$, which cancels the θ_{2n+1} term. Note that $c_1 \neq 0$ w.h.p., since $d_{2n} \neq 0$ with high probability and ω_{2n+1} contains $t_1 \cdot d_{2n}$.
(4) Compute an arbitrary vector $(d_2, \ldots, d_{n-1})$ satisfying $\delta_{eq} \cdot \omega_{n+1} + \delta_o \cdot \rho_{n+1} + \delta_{d2} \cdot \mu_{n+1} + \delta_{d1} \cdot \gamma_{n+1} + \delta_{\phi} \cdot \eta_{n+1} = 0$, which cancels the θ_{n+1} term.

Similarly as previous case, compute $\pi = \sum_{i=0, i \neq n+1}^{2n} g_i^{\theta_i}$ using $g, \{g_i\}_{i \in [2n] \setminus \{n+1\}}$ and return the simulated proof $\boldsymbol{\pi} = (\bar{C}_b, C_s, \pi)$.

Output : When $\mathcal{A}$ halts, it outputs a statement $(C, \Phi = (\phi_1, \ldots, \phi_m)$, for some $m \in [1, n]$, together with a verifying proof $\boldsymbol{\pi} = (\bar{C}_b, C_s, \pi)$.

In the algebraic group model, $\mathcal{A}$ will also provide the representation of each group element corresponding to all observed group elements in the same group. Specifically, $\mathcal{A}$ will give the representations of $\bar{C}_b$ w.r.t $(\bar{g}, \{\bar{g}_i\}_{i \in [n]})$ and the group elements $\{\bar{C}_b^{(i)}\}$ contained in simulated proofs $\{\boldsymbol{\pi}^{(i)}\}$. Likewise, $\mathcal{A}$ must provide a representation of C_s w.r.t $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$ and the group elements $\{C_s^{(i)}, \pi^{(i)}\}$ contained in simulated proofs $\{\boldsymbol{\pi}^{(i)}\}$. Note that for each $i \in Q_S$, $\mathcal{B}$ knows a representation of $(\bar{C}_b$ w.r.t $(\bar{g}, \{\bar{g}_i\}_{i \in [n]})$ and a representation of C_s w.r.t $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$. It also knows a representation of each $\pi^{(i)}$ w.r.t $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$. Therefore, from $\mathcal{A}$'s output and the random coins of the simulation, the extractor $\mathcal{E}$ can obtain scalars $\{(\theta_i, a_i, d_i)\}_{i \in [2n] \setminus \{n+1\}}$ and $\{c_i\}_{i \in [n]}$ such that

$$\pi = \prod_{i=0, i \neq n+1}^{2n} g_i^{\theta_i}, \quad C = \prod_{i=0, i \neq n+1}^{2n} g_i^{a_i} \quad \bar{C}_b = \prod_{i=0}^{n} \bar{g}_i^{c_i}, \quad C_s = \prod_{i=0, i \neq n+1}^{2n} g_i^{d_i}$$

If the representation of C, i.e., $(a_0, a_1, \ldots, a_n, a_{n+2}, \ldots, a_{2n})$, satisfies $a_1 \in \Phi$ and $a_i = 0$ for all $i \in [2, 2n] \setminus \{n+1\}$, then $\mathcal{E}$ can

simply output $(a_0, a_1) \in \mathbb{Z}_p^2$ as a valid opening of the Pedersen commitment C to a member $a_1 \in \Phi$.

Next, we consider the case where $a_1 \notin \Phi$ or there exists $i \in [2, 2n] \setminus \{n+1\}$ such that $a_i \neq 0$. Note that the left-hand side of (14) is obtained by raising the sub-verification equations (8)-(13) to the powers $\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi$ respectively, and then multiplying the results. Therefore, it can be written as $e(g, \bar{g})^{P_{agg}(\beta)}$, where the polynomial $P_{\text{agg}}[X]$ is defined as

$$\begin{aligned} P_{\text{agg}}[X] =& \delta_{eq} \cdot P_{eq}[X] + \delta_b \cdot P_b[X] + \delta_o \cdot P_o[X] + \\ & \delta_{d2} \cdot P_{d2}[X] + \delta_{d1} \cdot P_{d1}[X] + \delta_\phi \cdot P_\phi[X], \end{aligned} \tag{19}$$

which is a linear combination of following polynomials

$$\begin{aligned} P_{eq}[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{n} t_i \cdot s_i \cdot X^{n+1-i}\right) \\ &\quad - \left(\sum_{i=1}^{n} t_i \cdot X^i\right) \cdot \left(\sum_{i=0, i\neq n+1}^{2n} d_i \cdot X^i\right) = \sum_{i=0}^{3n} \omega_i \cdot X^i \\ P_b[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=0, i\neq n+1}^{2n} d_i \cdot X^i - \sum_{i=1}^{n} s_i \cdot X^{n+1-i}\right) \\ &= \sum_{i=0}^{3n} \varphi_i \cdot X^i \\ P_o[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{m} X^{n+1-i}\right) - X^{n+1} = \sum_{i=0}^{n+m} \rho_i \cdot X^i \\ P_{d2}[X] &= \left(\sum_{i=0, i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot X^{n-1} = \sum_{i=n-1}^{3n-1} \mu_i \cdot X^i \\ P_{d1}[X] &= \left(\sum_{i=0, i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot \left(\sum_{i=2}^{n} u_i \cdot X^{n+1-i}\right) = \sum_{i=0}^{3n} \gamma_i \cdot X^i \\ P_\phi[X] &= \left(\sum_{i=0}^{n} c_i \cdot X^i\right) \cdot \left(\sum_{i=1}^{m} \phi_i \cdot X^{n+1-i}\right) \\ &\quad - \left(\sum_{i=0, i\neq n+1}^{2n} a_i \cdot X^i\right) \cdot X^n = \sum_{i=0}^{3n-1} \eta_i \cdot X^i \end{aligned}$$

Let $P_{\text{agg}}[X] = \sum_{i=0}^{3n} \nu_i \cdot X^i$. Note that the verification equation (14) implies

$$\pi = \sum_{i=0}^{3n} g_i^{\nu_i}. \tag{20}$$

Since the extractor $\mathcal{E}$ knows the scalars $\{(a_i, d_i)\}_{i\in[2n]\setminus\{n+1\}}$ and $\{c_i\}_{i\in[n]}$, and can compute all the random values $\{(s_i, t_i)\}_{i\in[n]}$ and $(u_i)\}_{i\in[2,n]}$, $\mathcal{E}$ can recover $\{\nu_i\}_{i=0}^{3n}$. Additionally, $\mathcal{E}$ knows π's representation, $\pi = \prod_{i=0, i\neq n+1}^{2n} g_i^{\theta_i}$, which is revealed by $\mathcal{A}$ as part of its output. Let

$$R[X] = \sum_{i=0, i\neq n+1}^{2n} (\nu_i - \theta_i) \cdot X^i + \nu_{n+1} \cdot X^{n+1} + \sum_{i=2n+1}^{3n} \nu_i \cdot X^i. \tag{21}$$

We first consider the case where there exists $k \in [n+2, 2n]$ such that $a_k \neq 0$. We argue that if $a_k \neq 0$, then $\nu_{n-1+k} = 0$ cannot hold, except with negligible probability. This follows from the following reasoning:

- Let $\boldsymbol{\zeta}_{n-1+k} = (\omega_{n-1+k}, \varphi_{n-1+k}, \rho_{n-1+k}, \mu_{n-1+k}, \gamma_{n-1+k}, \eta_{n-1+k})$. Note that $\mu_{n-1+k} = a_k \neq 0$, so $\boldsymbol{\zeta}_{n-1+k} \neq \mathbf{0}$.
- If $\boldsymbol{\zeta}_{n-1+k} \neq \mathbf{0}$, then $\nu_{n-1+k} \neq 0$ with probability $1 - 1/p$ since $\boldsymbol{\delta} := (\delta_{eq}, \delta_b, \delta_o, \delta_{d2}, \delta_{d1}, \delta_\phi) = H_{\text{agg}}(C, \bar{C}_b, C_s)$ is derived *after* the choice of $\{(a_i, c_i)\}_{i\in[2n]\setminus\{n+1\}}$ and $\{d_i\}_{i\in[n]}$, which determines $\boldsymbol{\zeta}_{n-1+k}$. Thus, $\langle \boldsymbol{\delta}, \boldsymbol{\zeta}_{n-1+k} \rangle = 0$ with probability $1/p$ for a random $\boldsymbol{\delta}$.

For $k \in [n+2, 2n]$, if $\nu_{n-1+k} \neq 0$, where $n-1+k$ lies within the range $[2n+1, 3n-1]$, we are guaranteed that $R[X]$ defined in (21) is non-zero, then $\beta \in \mathbb{Z}_p$ can be found by $\mathcal{E}$ among the non-zero $R[X]$, which breaks the $(2n, n)$-dlog assumption.

Next, we discuss the case where either $\phi_\ell \in \Phi$ or $(a_2, \cdots, a_n) \neq \mathbf{0}^{n-1}$. From (19), we have

$$\begin{aligned} \nu_{n+1} =& \delta_{d1} \cdot \underbrace{\sum_{i=2}^{n} a_i \cdot u_i}_{\triangleq \gamma_{n+1}} + \delta_{eq} \cdot \underbrace{\sum_{i=1}^{n} t^i \cdot (c_i \cdot s_i - d_{n+1-i})}_{\triangleq \omega_{n+1}} \\ & + \delta_\phi \cdot \underbrace{(\sum_{i=1}^{m} c_i \cdot \phi_i - a_1)}_{\triangleq \eta_{n+1}} + \delta_o \cdot \underbrace{(-1 + \sum_{i=1}^{m} c_i)}_{\triangleq \rho_{n+1}} \\ & + \delta_b \cdot \underbrace{\sum_{i=1}^{n} (d_{n+1-i} - s_i) \cdot c_i}_{\triangleq \varphi_{n+1}} + \delta_{d2} \cdot \underbrace{a_2}_{\triangleq \mu_{n+1}} \end{aligned}$$

We argue that if $a_1 \notin \Phi := (\phi_1, \ldots, \phi_m)$ or $(a_2, \ldots, a_n) \neq \mathbf{0}^{n-1}$, we cannot have $\nu_{n+1} = 1$, except with negligible probability. This follows from the following arguments:

- $\boldsymbol{\zeta}_{n+1} := (\omega_{n+1}, \varphi_{n+1}, \rho_{n+1}, \mu_{n+1}, \gamma_{n+1}, \eta_{n+1}) = \mathbf{0}$ holds with negligible probability if $a_1 \notin \Phi$ or $(a_2, \ldots, a_n) \neq \mathbf{0}^{n-1}$. Specifically, we have $\mu_{n+1} \neq 0$ if $a_2 \neq 0$. Indeed, when $(a_2, \ldots, a_n) \neq \mathbf{0}^{n-1}$, $\gamma_{n+1} = 0$ with probability $1/p$ over the random choice of $\{u_i\}_{i=2}^n$ since $u_i = H_u(i, [2, n], C, \bar{C}_b, C_s)$ is derived uniformly *after* the choice of $\{a_i\}_{i=2}^n$. Likewise, when $d_{n+1-i} \neq c_i \cdot s_i$ for some $i \in [n]$, we have $\omega_{n+1} = 0$ with probability $1/p$ since $(t_1, \ldots, t_n) = H_t(\boldsymbol{s}, C, \bar{C}_b, C_s)$ is derived uniformly *after* the choice of $\boldsymbol{s}$, $\{d_i\}_{i\in[0,2n]\setminus\{n+1\}}$, and $\{c_i\}_{i\in[0,n]}$. Then, if $d_{n+1-i} = c_i \cdot s_i$ for all $i \in [n]$, we have $\varphi_{n+1} = \sum_{i=1}^n s_i \cdot (c_i - 1) \cdot c_i$, which cancels with probability $1/p$ if there exists $i \in [n]$ such that $c_i \notin \{0, 1\}$. To see this, we distinguish two cases:

  (1) If $\boldsymbol{s} = H_s(C, \bar{C}_b)$ was programmed in a simulation query, we only have φ_{n+1} with probability $1/p$. Specifically, in the simulation case where $a_{2n} \neq 0$, $\mathcal{B}$ chose $(c_1, \ldots, c_n)$ so that to have $\varphi_{n+1} = \sum_{i=1}^n s_i \cdot (c_i - 1) \cdot c_i = s_1 \cdot c_1 \cdot (c_1 - 1)$ with $s_1 \leftarrow\!\$\, \mathbb{Z}_p$ and $c_1 \notin \{0, 1\}$. While in the simulation case where $a_{2n} = 0$, $\mathcal{B}$ chose $(c_1, \ldots, c_n)$ so that to have $\varphi_{n+1} = \sum_{i=1}^n s_i \cdot (c_i - 1) \cdot c_i = s_1 \cdot c_1 \cdot (c_1 - 1) + s_n \cdot c_n \cdot (c_n - 1)$ with $s_1, s_n \leftarrow\!\$\, \mathbb{Z}_p$, $c_n = -t_n \leftarrow\!\$\, \mathbb{Z}_p$, and $c_1 \notin \{0, 1\}$.

(2) If $H_s(C, \bar{C}_b)$ was programmed in a simulation query, then $\boldsymbol{s} = H_s(C, \bar{C}_b)$ was defined after $\mathcal{B}$ obtained the representation $\{c_i\}_{i=0}^n$ of $\bar{C}_b$. Over the choice of $\boldsymbol{s}$, we have $\sum_{i=1}^n s_i \cdot (c_i - 1) \cdot c_i = 0$ with probability $1/p$.

If none of the above events occurs and $\rho_{n+1} = 0$, we have that there must be some $i \in [1, m]$ such that $c_i = 1$. If further there is $\eta_{n+1} = 0$, we have $a_1 = \sum_{i=1}^m c_i \cdot \phi_i$, $c_i \in \{0, 1\}$ for all $i \in [m]$, and there is some $i \in [1, m]$ such that $c_i = 1$, which contradicts the hypothesis $a_1 \notin (\phi_1, \ldots, \phi_m)$.

- If $\boldsymbol{\zeta}_{n+1} \neq \boldsymbol{0}$, then $\nu_{n+1} \neq 0$ with probability $1 - 1/p$ since $\boldsymbol{\delta} := (\delta_{\text{eq}}, \delta_{\text{b}}, \delta_{\text{o}}, \delta_{\text{d2}}, \delta_{\text{d1}}, \delta_{\phi}) = H_{\text{agg}}(C, \bar{C}_b, C_s)$ is derived *after* the choice of $\{(a_i, c_i)\}_{i \in [2n] \setminus \{n+1\}}$ and $\{d_i\}_{i \in [n]}$, which determines $\boldsymbol{\zeta}_{n+1}$. Thus, $\langle \boldsymbol{\delta}, \boldsymbol{\zeta}_{n+1} \rangle = 0$ with probability $1/p$ for a random $\boldsymbol{\delta}$.

If $\nu_{n+1} \neq 0$, we are guaranteed that $R[X]$ defined in (21) is non-zero. Similarly, $\beta \in \mathbb{Z}_p$ can be found by $\mathcal{E}$ among the non-zero $R[X]$, which breaks the $(2n, n)$-dlog assumption.

A.4 Security Analysis of Scheme in Sec 4.2

Since $(P_h[X], P_u[X], P_t[X])$, the real polynomials defined in (16), are publicly computable, the outsourcing computation process for $C_h = g^{P_h[X]}, \bar{C}_u = \bar{g}^{P_u[X]}, \bar{C}_t = \bar{g}^{P_t[X]}$ does not need to consider the zero-knowledge properties. The security analysis follows the proof of Theorem 4.2, with additional consideration of malformed $C_h, \bar{C}_u, \bar{C}_t$ in the proof generated by the adversary $\mathcal{A}$.

Let $(F_h[X], F_u[X], F_t[X])$ be the polynomials recovered from the representation of the commitments $C_h, \bar{C}_u, \bar{C}_t$ such that $C_h = g^{F_h[X]}, \bar{C}_u = \bar{g}^{F_u[X]}, \bar{C}_t = \bar{g}^{F_t[X]}$. Considering the case that at least one of $C_h, \bar{C}_u, \bar{C}_t$ is malformed, i.e., $(P_h[X], P_u[X], P_t[X]) \neq (F_h[X], F_u[X], F_t[X])$.

Note that the KZG verification Equation (17) implies

$$\delta_h \cdot (F_h[\beta] - p_h) + \delta_u \cdot (F_u[\beta] - p_u) + \delta_t \cdot (F_t[\beta] - p_t) - \pi(\beta) \cdot (\beta - z) = 0$$

where $\pi[X]$ is defined by the representation of $\pi_{\text{KZG}} = g^{\pi[\beta]}$.

From this, we construct the following polynomial $R'[X]$:

$$\begin{aligned} R'[X] =& \delta_h \cdot (F_h[X] - p_h) + \delta_u \cdot (F_u[X] - p_u) \\ &+ \delta_t \cdot (F_t[X] - p_t) - \pi(X) \cdot (X - z) \end{aligned} \tag{22}$$

We will demonstrate that if there exists a malformed input in $C_h, \bar{C}_u, \bar{C}_t$, then the extractor can compute $\beta \in \mathbb{Z}_p$ with non-negligible probability. Without loss of generality, we assume $F_h[X] \neq P_h[X]$. Since $z \leftarrow\!\!\$\, H(\Delta, C_h, \bar{C}_u, \bar{C}_t)$ is uniformly randomly chosen after $F_h[X], F_u[X], F_t[X]$ have been fixed, and the maximum degree of $F_h[X]$ is $2n$, it follows that the probability that the point z is a root of the non-zero polynomial $F_h[X] - P_h[X]$ is negligible. Specifically,

$$\Pr[F_h[z] - P_h[z] = 0] \leq \frac{2n}{p}.$$

Furthermore, since $(\delta_h, \delta_u, \delta_t) \leftarrow\!\!\$\, H(\Delta, C_h, \bar{C}_u, \bar{C}_t, z)$ is uniformly randomly chosen after $F_h[X], F_u[X], F_t[X]$ and z have been fixed, the probability that the point z is a root of the polynomial

$$\delta_h \cdot (F_h[X] - P_h[X]) + \delta_u \cdot (F_u[X] - P_u[X]) + \delta_t \cdot (F_t[X] - P_t[X])$$

is also negligible. Specifically, it is no greater than

$$\frac{2n}{p} + \left(1 - \frac{2n}{p}\right) \cdot \frac{1}{p} \leq \frac{2n+1}{p}.$$

Therefore, according to the Factor Theorem, the probability that the adversary provides a polynomial $\pi(X)$ such that $R'[X] = 0$ is no greater than $\frac{2n+1}{p}$. This implies that with overwhelming probability, the extractor can compute $\beta \in \mathbb{Z}_p$ from a non-zero $R'[X]$.

B Security Definitions of Scoped Linkable Ring Signatures

Here, we provide the formal security definitions of scoped LRS, including anonymity, scoped linkability, and non-slanderability. The following oracles are defined for security experiments:

- KGO() $\rightarrow$ pk: Generates a key pair (pk, sk) and returns pk.
- CO(pk) $\rightarrow$ sk: Returns the private key sk associated with pk.
- SO(sc, R, pk, m) $\rightarrow \sigma$: Provides a valid signature σ on (m, sc) for ring R and pk $\in R$.

We now define anonymity, which requires that the adversary cannot determine which specific key was used in the ring signature. A key distinction from "standard" LRS schemes (i.e., those with full message linkability) is that, even if the adversary has access to a signing oracle for the challenged two public keys. In standard LRS schemes, such access would enable the adversary to easily win. In scope LRS schemes, however, the adversary can only win easily if they use the same scope in the signing oracle. Therefore, we describe a stronger anonymity game by allowing access to signing oracles with scopes different from the challenge scope.

Definition B.1 (Anonymity). A LRS scheme is anonymous if no adversary $\mathcal{A}$ can distinguish the signer within the ring. The anonymity game $\text{Game}_{\text{anon}}$ between a challenger $\mathcal{C}$ and $\mathcal{A}$ is defined as:

- $\mathcal{C}$ runs $\text{Setup}(1^\lambda, 1^n)$ to generate pp and initializes oracles.
- $\mathcal{A}$ interacts with KGO, CO and SO, and outputs a ring R, a message m, an scope identifier sc and two indices i_0, i_1.
- $\mathcal{C}$ picks $b \in \{0, 1\}$ and computes $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}_{i_b}, \text{m}, \text{sc}, R)$ and sends σ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess b'.
- $\mathcal{A}$ wins if:
  - $b' = b$.
  - R produced by KGO, $\text{pk}_{i_0}, \text{pk}_{i_1} \in R$ and $\text{pk}_{i_0} \neq \text{pk}_{i_1}$.
  - $\text{pk}_{i_0}, \text{pk}_{i_1}$ were never queried to CO.
  - (pk_{i_0}, sc) and (pk_{i_1}, sc) were never queried to SO.

The scheme is anonymous if the advantage of $\mathcal{A}$, denoted as $\text{Adv}^{\text{LRS}}_{\text{anon}}(\mathcal{A}) = \left|\Pr[b' = b] - \frac{1}{2}\right|$, is negligible.

Definition B.2 (Scoped Linkability). A LRS scheme ensures that no adversary $\mathcal{A}$ can produce two unlinkable signatures using the same private key for the same scope [23]. The scoped linkability game $\text{Game}_{\text{link}}$ is defined as:

- $\mathcal{C}$ runs $\text{Setup}(1^\lambda, 1^n)$ to generate pp and shares it with $\mathcal{A}$.
- $\mathcal{A}$ interacts with KGO, CO, and SO and outputs the signatures $\{(\text{sc}, \sigma_k, \text{m}_k, R_k)\}_{k=1}^q$.
- $\mathcal{A}$ wins if:
  - $\text{Verify}(\text{pp}, \sigma_k, \text{m}_k, \text{sc}, R_k) = 1$ for all $k \in [q]$,
  - $\text{Link}(\text{pp}, \text{sc}, \sigma_{k_1}, \sigma_{k_2}) = 0$ for $k_1, k_2 \in [q]$ and $k_1 \neq k_2$,

– R_k is produced by KGO, no σ_k is produced by SO for all $k \in [q]$, and CO is queried $< q$ times.

The scheme is scoped linkable if $\mathsf{Adv}^{\mathsf{LRS}}_{\mathsf{link}}(\mathcal{A})$, the probability of winning $\mathsf{Game}_{\mathsf{link}}$, is negligible.

Definition B.3 (Non-slanderability). A LRS scheme ensures non-slanderability if no adversary $\mathcal{A}$ can produce a signature that falsely links to an honest signer. The non-slanderability game $\mathsf{Game}_{\mathsf{slan}}$ is defined as:

- $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda, 1^n)$ to generate pp and shares it with $\mathcal{A}$.
- $\mathcal{A}$ selects an scope identifier sc, a message m, a ring R and a public key $\mathsf{pk} \in R$. and send them to $\mathcal{C}$. Then $\mathcal{C}$ runs $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m}, \mathsf{sc}, R) \rightarrow \sigma$ where sk is the corresponding to pk, and return σ to $\mathcal{A}$.
- $\mathcal{A}$ interacts with KGO, CO, and SO and outputs $(\sigma', \mathsf{m}', R')$.
- $\mathcal{A}$ wins if:
  – $\mathsf{Verify}(\mathsf{pp}, \sigma', \mathsf{m}', \mathsf{sc}, R') = 1$, and $\mathsf{Link}(\mathsf{pp}, \mathsf{sc}, \sigma, \sigma') = 1$
  – R, R' produced by KGO, pk has not been queried by CO and SO, σ' was not produced by SO.

The scheme is non-slanderable if $\mathsf{Adv}^{\mathsf{LRS}}_{\mathsf{slan}}(\mathcal{A})$, the probability of winning $\mathsf{Game}_{\mathsf{slan}}$, is negligible.

C Analysis of the MTree + SNARK Approach and Comparison with Our Work

In this section, we first present the MTree + SNARK approach for generating constant-size linkable ring signatures, outlining its construction steps. We then analyze the verification cost of this approach in the ad hoc ring setting. Finally, we present a direct comparison with our proposed scheme, supported by both theoretical analysis and experimental results, to highlight the efficiency advantages of our design.

The MTree + SNARK approach for constant-size LRS involves the following workflow:

(1) **Signature Generation:** The signer first generates a standard signature on the message using a ZK-friendly signature scheme (e.g., Schnorr signatures).
(2) **Merkle Tree Construction:** The signer temporarily selects n public keys to form the ring and organizes them into a Merkle tree by using ZK-friendly hash function (e.g. MiMC)„ where each leaf node corresponds to one public key in the ring.
(3) **Proof Generation (Constant-size SNARK):** The signer computes a SNARK proof that proves two facts in one shot: (a) correctness of the generated signature; and (b)knowledge of a valid Merkle path (as a private input) from his public key (a leaf, private input) to the root (public input) of the Merkle tree.
(4) **Verification:** The verifier checks the SNARK proof, validating both the correctness of the signature and the inclusion of the public key within the ring represented by the Merkle root. To ensure soundness, the verifier must check that the Merkle root correctly corresponds to the claimed ring-an operation that incurs $O(n)$ cost.

Here we elaborate on the $O(n)$ cost involved in verifying the correctness of the Merkle root below. There are two possible ways to perform this check. One is to verify the Merkle root *inside* the SNARK circuit by supplying the entire ring and the root as public inputs. This results in $O(n)$ public inputs and thus $O(n)$ verification cost, in addition to significantly increasing proving time. The other, more practical approach is to verify the Merkle root *outside* the SNARK, requiring the verifier to recompute the root directly from the ring. This also incurs $O(n)$ cost. In the following we adopt this latter option for the check.

Verification cost analysis. We now analyze the verifier's computational cost. In addition to the constant number of pairings and group exponentiation operations from SNARK verification, the verifier must perform $O(n)$ MiMC hashes in the MTree + SNARK construction. In our experimental setup (BN254 curve, 128-bit security, exponent $e = 7$, and 91 rounds for MiMC, following recommended parameters in [2] for BN254), a single MiMC hash entails 91 field additions and 364 field multiplications. Benchmarks averaged over 10^6 iterations show that one MiMC hash takes 15.120 µs, while one field multiplication costs 31.684 ns, yielding a ratio of roughly $15.120\,\mu s / 31.684\,\text{ns} \approx 455.4\times$, which aligns with expectations. Indeed, a single MiMC hash is of the same order of magnitude as a $\mathbb{G}_1$ exponentiation, which we measured at 67.743 µs.

C.1 Comparison with Our Proposed Approach

In contrast to the MTree + SNARK method, our approach limits the use of expensive operations (e.g., group exponentiation, pairing, MiMC hashing) to a constant number, regardless of the ring size. As a result, our verification complexity remains essentially constant, which leads to significantly better performance compared to the MTree + SNARK approach.

Theoretical comparison. The MTree + SNARK verifier requires the computation of n MiMC hashes, each of which is equivalent to approximately 455 field multiplications (as established in our experimental benchmarks). In total, this results in about $455n$ field-multiplication equivalents for verification. By contrast, our scheme requires only $3n$ field multiplications. This yields a predicted gap factor of roughly $455/3 \approx 151\times$ as the ring size increases.

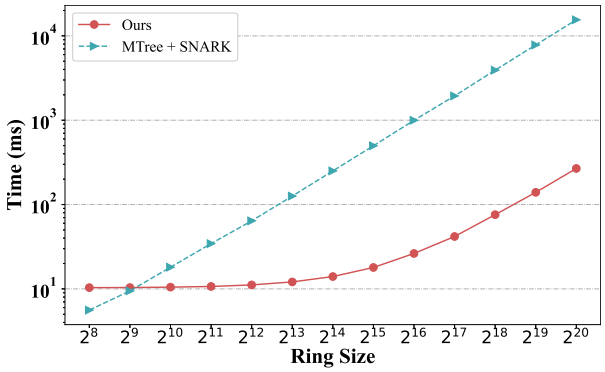


Figure 2: Verification time for the MTree + SNARK method and our scheme.

Experimental Validation. To concretely illustrate these differences, we implemented both approaches under identical settings (Groth16 as the SNARK, MiMC as the ZK-friendly hash, and Schnorr as the ZK-friendly signature scheme). The code is publicly available[11]. We measured both signing and verification costs, and the

[11] https://github.com/TuZhengzhou/lrs-v2.git

results are shown in Figure 2 and Figure 3, which clearly highlight the advantages of our method:

- For ring size 2^{10}, the MTree + SNARK approach required 794.25 ms for signing and 18.07 ms for verification, whereas our method needed only 235.20 ms for signing and 10.41 ms for verification.
- For ring size 2^{15}, verification times were 497.38 ms (MTree + SNARK) versus 17.94 ms (our approach), providing a 27.7× improvement.
- At larger scales (e.g., 2^{17}), the MTree + SNARK method required 1.940 s for verification, whereas our verifier completed in just 41.79 ms—a practical improvement factor of approximately 46.4×.

In summary, while the MTree + SNARK method provides constant-size signatures and efficient signing, it does not achieve near-constant-time verification in ad hoc rings due to fundamental limitations shared with other accumulator-based methods. Both theoretical and experimental results confirm our conclusion: the verifier cost of our design significantly outperforms the MTree + SNARK approach, particularly for large ring sizes, achieving practical near-constant-time verification (the goal of this paper) while maintaining excellent signing efficiency.

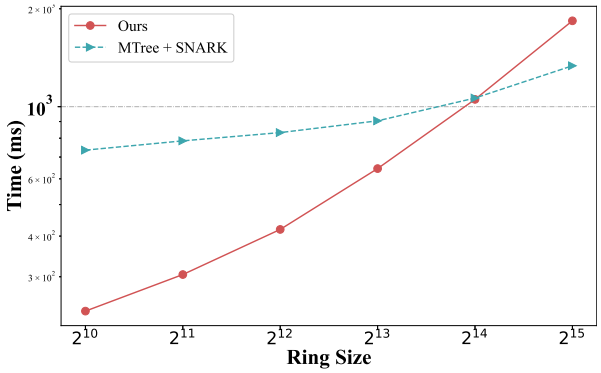


Figure 3: Signing time for the MTree + SNARK method and our scheme.