Anonymity-Enhancing Multi-Hop Locks for Monero-Enabled Payment Channel Networks

Xiaohu Wang, Chao Lin, Xinyi Huang, *Member, IEEE*, and Debiao He, *Member, IEEE*

***Abstract*— Payment Channel Networks (PCNs) are innovative second-layer scaling technologies that aim to improve transaction rates, reduce on-chain storage costs, and enable efficient atomic swaps for blockchain-based cryptocurrencies. Despite offering features like relationship anonymity, scriptless script, and cross-chain fairness, current PCNs encounter challenges in achieving identity anonymity and maintaining the fungibility of cryptocurrency units. PayMo, proposed in ESORICS'22, addresses payment anonymity but is limited to Monero, posing difficulties in extending it to a PCN framework. In response, this paper presents a novel Anonymity-Enhancing Multi-Hop Locks (AEMHL) mechanism for Monero-enabled PCNs. The AEMHL mechanism leverages our generic Linkable Ring Adaptor Signature (LRAS) construction and a minimalist PCN framework called anonymous multi-hop locks. This approach effectively combines privacy protection and simplicity while ensuring Monero's fungibility without the need for specialized scripting support. Security properties, including atomicity, consistency, and anonymity-enhancement, are demonstrated using a universal composability model. Additionally, two optimized LRAS-based schemes are proposed to accommodate multi-hop locks construction in diverse scenarios. Through rigorous security analysis and performance evaluation, we confirm that AEMHL meets essential security objectives and provides efficient and practical solutions for privacy-conscious users within PCNs.**

***Index Terms*— Blockchain, payment channel network, multi-hop payment protocol, identity anonymity, Monero, universal composability.**

I. Introduction

BLOCKCHAIN, as a decentralized platform for distributed operation, is changing traditional business logic and institutional operation mode. It has been widely applied in finance, Internet of Things, medical health, and various other fields. For example, the blockchain empowers cryptocurrencies driven by smart contracts, which reduces the need for intermediaries in traditional currency transactions and improves transaction soundness to a certain extent. Despite the potential of cryptocurrencies, a significant drawback is their low scalability, which results in a low transaction rate. For instance, Bitcoin now allows for 10 transactions per second (tps) and demands up to an hour for confirmation. These limitations significantly hinder the widespread application and promotion of blockchain technology.

Prompted by the above scalability issues, the exploration of off-chain payment solutions arises, where users can make multiple local transactions before only registering the final balance on the blockchain. *Payment Channels* (PC) and its extension *Payment Channel Networks* (PCN) are two popular off-chain payment solutions that have shown great potential for improving scalability. The PC protocol involves two users creating a two-parties payment channel with just a single on-chain transaction, and then being able to make multiple payments locally by adjusting the channel's balance without recording them on the blockchain. Finally, a closing transaction will be recorded on the blockchain if either party would like to close the channel.

PCN builds on the concept of PC and enables payments between nodes who don't share a direct PC. Instead, PCN creates a chain of intermediate nodes that connect the receiver and the sender, known as a payment path. This allows PCN to achieve secure and efficient cross-currency payments, including atomic swaps. The initial proposal for PCN relied on a scripting functionality of blockchain dubbed *Hash Time-Lock Contracts* (HTLC) [1], [2]. It enables payments throughout the path with successful completion based on releasing a pre-image of a hash value within a specific time limit. However, the use of HTLC is limited to blockchains that have advanced scripting capabilities, and it also faces on-chain privacy concerns due to the potential linking of each payment transaction. In addition, there are also security issues related to payment paths, such as the possibility of wormhole attacks [3].

To address the issues with HTLC, Malavolta et al. [3] proposed *Anonymous Multi-Hop Locks* (AMHL), which ensures relationship anonymity without the need for special scripts (known as scriptless scripts [4]). The scheme leverages the algebraic structure of ECDSA and Schnorr signatures, along with a crucial homomorphic one-way function, to construct a cascading mechanism. It allows the two parties within each channel to establish a lock similar to onion routing [5], ensuring that they don't learn locking information concerning the other channels in an AMHL except for its immediate neighbors. This provides a solution to both privacy leakage and script dependency issues. Unfortunately, these techniques appear inadequate for extending to PCN protocols based on other prominent signature schemes, such as ring signatures [6] and its variants [7], which offer identity anonymity properties. With the potential for these signature schemes to facilitate efficient identity anonymization, it is

Manuscript received 26 July 2023; revised 16 November 2023 and 17 December 2023; accepted 19 December 2023. Date of publication 22 December 2023; date of current version 12 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62032005, Grant 62102089, Grant U21A20466, Grant 62325209, Grant 62102050, Grant 61972094, Grant 62202226, and Grant 62272104; and in part by the Fundamental Research Funds for the Central Universities under Grant 30922010917. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Paolo Gasti. *(Corresponding author: Chao Lin.)*

Xiaohu Wang is with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China (e-mail: wangxiaohu49@163.com).

Chao Lin is with the College of Information Science and Technology, Jinan University, Guangzhou 510632, China (e-mail: linchao91@fjnu.edu.cn).

Xinyi Huang is with the Artificial Intelligence Thrust, Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511455, China (e-mail: xinyi@ust.hk).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shandong Provincial Key Laboratory of Computer Networks, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China (e-mail: hedebiao@163.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2023.3346177, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2023.3346177

likely that more cryptocurrencies will adopt them in the future. Furthermore, although AMHL successfully achieves basic relationship anonymity, it appears to overlook the issue of identity anonymity between users on the payment path. This may raise concerns regarding the exposure of personal information and the risk of unauthorized access to sensitive data.

Thyagarajan et al. [8] proposed a scriptless locking mechanism in response to the above compatibility issues faced by AMHL. This mechanism is compatible with most signatures and provides on-chain privacy in the payment path. The mechanism utilizes a lockable signature, which cascades the previous channel's signature hash and the next channel's signature hash in a scale-like manner. This lockable signature ensures compatibility with most 'special' structure signatures, and the cascade form and multiple protocol interactions of the sender ensure the security of the payment path and privacy on the chain. However, this mechanism necessitates the sender's participation in multiple interactions, which increases the cost of channel initialization on the chain and the complexity of off-chain operations. Additionally, it still does not achieve the aforementioned identity anonymity.

Notably, certain existing PC protocols such as PayMo [9] achieve user identity anonymity by seamlessly adapting to Monero. This implementation relies on the property of the *Linkable Ring Signature* (LRS) [7], allowing for anonymous message signing within a ring of dynamically selected potential signers based on the signer's public key. Although interesting, extending PayMo to PCNs still lacks a locking mechanism that allows cascading channels and circumvents on-chain scripts. Through this mechanism, intermediate nodes can be constrained from cheating to obtain transfers in advance.

A. Challenges and Attempts

In summary, we emphasize the necessity of the Monero-oriented payment channel network we have constructed, along with the challenges encountered in the process. These challenges include incomplete identity anonymity, issues with token fungibility, and low protocol adaptability. First and foremost, it is important to note that most existing payment channel networks are primarily designed for mainstream cryptocurrencies. They often fail to address the privacy issues arising from transaction correlations between addresses on the blockchain. For example, adversaries can exploit transaction records associated with specific addresses to discern the true identities of transaction partners. Hence, the need to build a payment channel network tailored for the largest anonymous cryptocurrency, Monero, is our primary design objective. During the construction of PCN, the specific challenges we encountered are as follows:

1) Incomplete Identity Anonymity: This represents the core issue that our protocol aims to resolve. Current payment channel networks often assume that each pair of channel users on the payment link possesses a public address (shared public key) and its corresponding private address (personal public key). However, these networks typically only restricts intermediate nodes to collect user set information of neighbor nodes including personal public keys (i.e. relational anonymity). This approach doesn't prevent malicious users from deducing the parties ivolved in each channel through shared public keys. To address this, our new multi-hop payment protocol employs the proposed linkable ring signature to achieve complete user identity anonymity.

2) Impaired Fungibility: Fungibility requires that any third-party observers cannot distinguish transactions in the payment channel network from standard transactions on the underlying blockchain. Existing payment channel networks, such as the Bitcoin Lightning Network, establish channels by transferring coins to P2WSH 2-of-2 multi-signature addresses, but that make them easily identifiable. In contrast, our off-chain transactions follow construction methods similar to those used in Monero. This includes using anonymous sets to obscure signature-user pairs during the lock phase and employing commitments and zero-knowledge proofs to hide and verify secret information during the transaction transmission process.

3) Low Protocol Adaptability: To facilitate future work related to currency exchange, such as atomic swaps, it is essential to have a highly adaptable protocol. Existing payment channel network protocols are often limited to specific signature designs. For example, the Bitcoin Lightning Network is only compatible with ECDSA-based designs, and AMHL can adapt to tokens based on ECDSA and Schnorr signatures but lacks a built-in ring structure. Therefore, Our protocol aims to provide a efficient anonymous system that can be widely adaptable.

B. Our Contribution

In response to the above problems, this paper is dedicated to designing a novel anonymity-enhancing multi-hop locking mechanism for Monero-enabled PCNs. The protocol has a rosy trade-off in terms of token *fungibility*, privacy protection, and protocol performance, and can further optimize performance through a series of improved signature schemes. Our main contributions are presented as follows.

- We construct the first *Anonymity-Enhancing Multi-Hop Locks* (AEMHL) (Section IV-A) for Monero-enabled PCNs based on our proposed generic construction of *Linkable Ring Adaptor Signature* (LRAS). AEMHL guarantees meaningful identity anonymity and account security (like wormhole resistance) within a minimal PCN framework, which is comparable to the most advanced proposal [3], [8]. Our solution also is designed to be compatible with most signature programs without destroying Monero's fungibility. Moreover, AEMHL can replace the functionality of *Hash-lock*, eliminating the need for specific assumptions about the script underlying blockchain.
- We further optimize the general LRAS to reap two improved schemes, *Linkable DualRing Adaptor Signature* (LDRAS) and *Linkable DualRing Adaptor Signature Plus* (LDRAS+), each with independent interest (Section V). LDRAS features a dual-ring signature structure, which reduces computational complexity and overhead compared to the original structure. Meanwhile, LDRAS+ leverages the *Non-Interactive Sum Argument of knowledge* (NISA) [10] algorithm to achieve "shorter signature, faster verification". Specifically, LDRAS+ reduces the signature length of the LRAS from linear to logarithmic, significantly reducing the computational overhead of the verification algorithm. These improvements enhance the efficiency and practicality of the locking mechanism in various applications.
- We provide theoretical analysis and conduct experiments utilizing the *Multi-precision Integer and Rational Arithmetic C/C++ Library* (MIRACL) in various environments. Our results demonstrate that the performance of multiple instantiation schemes of our AEMHL is within the expected range. Furthermore, our optimized signature schemes demonstrate their versatility by adapting to different application scenarios when building a new PCN.

C. Organization

Section II provides an overview of the relevant literature on PC and PCN, as well as diverse signatures, and Section III introduces the preliminaries. Section IV presents AEMHL together with its security analysis, followed by the improved LDRAS and LDRAS+ schemes and security proofs of the former in Section V. Section VI compares and analyzes the performance of each scheme. Finally, the last section provides the conclusion of the paper.

II. Related Work

In this section, we present a literature overview of PC and PCN, as well as the diverse signature schemes utilized in their construction.

A. PC/PCN

As the second-layer extension technology of the blockchain, a series of PC and PCN [1], [2], [3], [8], [9] have been widely proposed and deeply studied. In 2016, the concept of HTLC was presented as a fundamental building block for the Lightning Network [1], [2]. This locking mechanism has been widely utilized in various blockchain applications [11], [12], [13]. However, these protocols, although universal to some extent, still rely on blockchain-specific scripts and cannot achieve scriptless scripts.

As mitigation, Malavolta et al. [3] proposed an alternative locking mechanism called AMHL, which provides enhanced on-chain privacy without relying on HTLC. Nonetheless, this protocol is specifically designed for transaction schemes using ECDSA and Schnorr signatures and does not achieve identity anonymity. Then, Thyagarajan et al. [9] proposed a payment channel protocol called PayMo, which guarantees perfect user identity anonymity. Nevertheless, PayMo still relies on certain underlying blockchain scripts and does not appear to be extended to PCN. To eliminate the dependency on scripts, Thyagarajan et al. [8] presented a scriptless locking mechanism with broader signature compatibility to address the narrow signature adaptation of AMHL. This protocol also fails to achieve identity anonymity, and its increased adaptability introduces higher computational overhead and on-chain costs.

B. Diverse Signatures

To enable the construction of PC and PCN with various anonymity properties, a range of related signature schemes [6], [9], [14], [15], [16], [17], [18] have been proposed as their fundamental building blocks. In 2002, Abe et al. [6] proposed the first ring signature structure with identity anonymity properties. This structure has been applied in the various construction of PC to protect payment path privacy and maintain the fungibility of cryptocurrencies like Monero. For example, the proposal [9] introduced a modified LRS called *Verifiable Timed Linkable Ring Signature* where signatures could stay secret for a set amount of time while still being verifiable. While these signatures provide privacy protection, their applicability is limited to anonymous cryptocurrencies.

To construct a more versatile locking mechanism, researchers started adopting adaptor signatures [14], which involve hard relation pairs. Aumayr et al. [15] introduced a PCN structure that is more efficient than the Lightning Network by leveraging adaptor signatures. This structure exploits the unique hard relation within the adaptor signature to design a novel mechanism for revoking old state channels, thereby avoiding the revocation overhead associated with the number of parallel payment conditions in the Lightning Network. Sui et al. [18] proposed a CLRAS that intends to provide identity anonymity. Unfortunately, this scheme only formally sign and authenticate messages against a set of public key, and fail to achieve the claimed anonymity and linkability.

III. Preliminaries

Throughout this paper, we adopt the following notations: λ represents the security parameter, $1^\lambda \in \mathbb{N}^+$ is the security binary string, $\text{negl}(\lambda)$ is a negligible function concerning λ, and Δ denotes a suitable randomness space defined by the algorithm. In addition, we represent the uniformly random sampling of an element from a set S by the symbol $x \xleftarrow{\$} S$ when given a set S.

A. Type-T Signature and Its Canonical Identification

1) Type-T Signatures: The Type-T signatures [6] typically consists of the four algorithms listed below:

1) $\text{SETUP}(\lambda) \rightarrow params$: This algorithm receives a security parameter λ as input, and generates global public parameters $params$ as outputs.
2) $\text{KEYGEN}(params) \rightarrow (pk, sk)$: This algorithm is executed by the certificate authority (CA). It receives global public parameters $params$ as inputs, and outputs a public-private key pair (pk, sk).
3) $\text{SIGN}(sk, M) \rightarrow \sigma$: This algorithm is executed by the signer. It receives the private key sk of the signer and a message M to be signed as inputs, and finally outputs the signature σ.
4) $\text{VERIFY}(pk, \sigma, M) \rightarrow \{0, 1\}$: This algorithm is executed by the verifier. It receives a message M, the public key pk of the signer and a signature σ of M as inputs, and outputs 1 if σ is valid on M under pk. Otherwise, it outputs 0.

The specific algorithm details of the Type-T signatures are outlined in Algorithm 1. Examples of its implementation include the Schnorr signatures [19], Guillou-Quisquater signatures [20] are presented as follows.

2) Schnorr Instantiation [19]: In the Schnorr instantiation, if the DL assumption holds, we state that the relation $A(r) = g^r$ constitutes a hard relation. Considering $\otimes$ as modular addition in a cyclic group and $\odot$ as multiplication, the following relations holds: 1) $A(r_1) \odot A(r_2) = g^{r_1} \cdot g^{r_2} = g^{r_1+r_2} = A(r_1 \otimes r_2)$; 2) $A(-r) = g^{-r} = (g^r)^{-1} = (A(r))^{-1}$; 3) $Z(sk, r_1, c) \otimes r_2 = r_1 + c \cdot sk + r_2 = (r_1 + r_2) + c \cdot sk = Z(sk, r_1 \otimes r_2, c)$; 4) $V(pk, z, c) = g^z \cdot pk^c = A(z) \odot V'(pk, c)$.

3) GQ Instantiation [20]: In the GQ instantiation, if the standard RSA assumption holds, we state that the relation $A(r) = r^e$ is a hard relation. Considering $\otimes$ as modular addition in a cyclic group and $\odot$ as multiplication, the following relations holds: 1) $A(r_1) \odot A(r_2) = r_1^e \cdot r_2^e = (r_1 \cdot r_2)^e = A(r_1 \otimes r_2)$; 2) $A(r^{-1}) = r^{-e} = (r^e)^{-1} = A(r)^{-1}$; 3) $Z(sk, r_1, c) \otimes r_2 = r_1 \cdot sk^c \cdot r_2 = (r_1 \cdot r_2) \cdot sk^c = Z(sk, r_1 \otimes r_2, c)$; 4) $V(pk, z, c) = z^e \cdot pk^c = A(z) \odot V'(pk, c)$.

4) Type-T Canonical Identification [21]: As a public-key authentication protocol with three-move operating, the Canonical identification is precisely defined in [6]. Notably, when the Fiat-Shamir transformation is applied to Type-T canonical identification, it results in a Type-T signature.

Definition 1: Assuming there are no Probabilistic Polynomial Time ($\mathcal{PPT}$) adversary $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{imp}(\lambda)$ is non-negligible, then a Type-T canonical identification is

Algorithm 1 Type-T Signature

Procedure S ETUP(λ) :
  return $params$;
Procedure K EYGEN($params$) :
  return (pk, sk);
Procedure S IGN(sk, M) :
  $r \xleftarrow{\$} \Delta_r$, $R = A(r)$;
  $c = H(M, R)$;
  $z = Z(sk, r, c)$;
  return $\sigma = (z, c)$;
Procedure V ERIFY(pk, σ, M) :
  parse $\sigma = (z, c)$;
  $R' = V(pk, z, c)$;
  if $c \neq H(M, R')$ **then**
    return 0;
  return 1;

Algorithm 2 Experiment $\mathbf{aWitExt}_{\mathcal{A},\prod_{\mathbf{Re},\Sigma}}$

Procedure: $\mathbf{aWitExt}_{\mathcal{A},\prod_{R_e,\Sigma}}(\lambda)$:
  $\Omega = \emptyset$; $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in [1, n]$;
  $\boldsymbol{pk} = \{pk_1, pk_2, \cdots, pk_n\}$;
  $(M^*, Y) \leftarrow A^{O_s, O_{ps}}(\boldsymbol{pk})$;
   $\hat{\sigma} \leftarrow \text{PreSign}(\boldsymbol{pk}, sk_j, Y)$;
  $\sigma^* \leftarrow A^{O_s, O_{ps}}(\hat{\sigma})$; $y^* \leftarrow \text{Ext}(Y, \sigma^*, \hat{\sigma})$;
  return $((M^* \notin \Omega) \wedge (Y, y^*) \notin$
   $R_e \wedge \text{Verify}(\boldsymbol{pk}, \sigma^*, M^*) = 1)$;
  Procedure: $O_s(M)$:
    $\sigma \leftarrow \text{Sign}(sk_j, M)$; $\Omega = \Omega \cup \{M\}$;
    return σ;
  Procedure: $O_{ps}(M, Y)$:
    $\hat{\sigma} \leftarrow \text{PreSign}(\boldsymbol{pk}, sk_j, Y, M)$; $\Omega = \Omega \cup \{M\}$;
    return $\hat{\sigma}$;

secure against impersonation under key-only attack. Where $\boldsymbol{Adv}^{imp}_{\mathcal{A}}(\lambda)$ *is defined as follows:*

$$\begin{aligned}\mathbf{Adv}^{imp}_{\mathcal{A}}(\lambda) := \Pr[&\text{VERIFY}(pk, z^*, c_{i^*}) \\ &= 1 | params \leftarrow \text{SETUP}(\lambda), \\ &(pk, sk) \leftarrow \text{KEYGEN}(params), (c_{i^*}, z^*) \\ &\leftarrow \mathcal{A}^{\text{CH}(\cdot)}(params, pk)].\end{aligned}$$

The oracle provides $\mathcal{A}$ *with the value* c_i *during the* i*-th query CH(*R_i*), where* $i^* \in [1, q_c]$ *and* q_c *represents the total number of queries made to the CH oracle.*

B. Formal Definition of LRAS and Security Models

If the following conditions holds, a relation R_e with a language $L_R = \{Y | \exists y : (Y, y) \in R_e\}$ is stated hard: (i) There exists a $\mathcal{PPT}$ generator, LockGen(λ), that outputs $(Y, y) \in R_e$; (ii) The probability of adversary $\mathcal{A}$ computing witness y is negligible for every $\mathcal{PPT}$ algorithm $\mathcal{A}$ when given $Y \in L_R$.

A linkable ring adaptor signature scheme $\prod_{R_e,\Sigma}$ is defined with respect to a hard relation R_e and a signature scheme Σ.

Definition 2 (Linkable Ring Adaptor Signatures): A linkable ring adaptor signature scheme $\prod_{R_e,\Sigma}$ *consists of eight algorithms (Setup, KeyGen, PreSign, PreVerify, Adapt, Verify, Ext, Link) defined below.*

- Setup(λ): This algorithm receives a security parameter λ as input, outputs system parameters $params$ including a hash function $H : \{0, 1\}^* \rightarrow \Delta_c$.
- KeyGen($params$): The algorithm is executed by the CA. It receives system parameters $params$ as inputs, and generates the public-private key pair (pk, sk) of signer as outputs.
- PreSign($\boldsymbol{pk}$, sk_j, Y, M): The algorithm is executed by the signer. It receives the set of public keys $\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}$ for the user group, private key sk_j (j is the index of the signer in user group), a statement $Y \in L_R$ and a message $M \in \{0, 1\}^*$, as inputs, and outputs a pre-signature $\hat{\sigma}$.
- PreVerify(Y, $\boldsymbol{pk}$, $\hat{\sigma}$, M): This algorithm is executed by the verifier. It receives a statement $Y \in L_R$, a message M to be signed, the set of public keys $\boldsymbol{pk}$ and a pre-signature $\hat{\sigma}$, and finally outputs a bit b.
- Adapt(M, $\hat{\sigma}$, $\boldsymbol{pk}$, (Y, y)): This algorithm is executed by the adapter. It received a message M, a hard relation pair (Y, y), the set of public keys $\boldsymbol{pk}$ and a pre-signature $\hat{\sigma}$ as inputs, and finally outputs a formal-signature σ.
- Verify($\boldsymbol{pk}$, σ, M): This algorithm is executed by the verifier. It received a formal signature σ, the set of public keys $\boldsymbol{pk}$ and a message M as inputs, and finally outputs 1 if the formal signature is valid. Otherwise, it outputs 0.
- Ext(Y, σ, $\hat{\sigma}$): This algorithm is executed by the extractor. It received a statement $Y \in L_R$, a signature σ and a pre-signature $\hat{\sigma}$ as inputs, and outputs a witness y such that $(Y, y) \in R$, or $\perp$.
- Link($\boldsymbol{pk}$, M', M'', σ', σ''): This algorithm is executed by the third-party(pre-signed and officially signed owners). It received a set $\boldsymbol{pk}$ and two message/signature pairs (M', σ') and (M'', σ'') as inputs, and outputs "Unlinked" or "Linked".

Definition 3 (Pre-signature Adaptability): Assuming the following condition holds for each message M *to be signed in the message space* Ω*, all pre-signatures* $\hat{\sigma}$*, and every hard relation pair* $(Y, y) \in R_e$*, then a linkable ring adaptor signature scheme* $\prod_{R_e,\Sigma}$ *satisfies pre-signature adaptability:*

$$\Pr\left[\text{Verify}(\boldsymbol{pk}, \sigma, M) = 1 \;\middle|\; \begin{array}{l} params \leftarrow \text{Setup}(\lambda), \\ (pk, sk) \leftarrow \text{KeyGen}(params), \\ \hat{\sigma} \leftarrow \text{PreSign}(\boldsymbol{pk}, sk, Y, M) \\ \text{PreVerify}(Y, \boldsymbol{pk}, \hat{\sigma}, M) = 1, \\ \sigma \leftarrow \text{Adapt}(M, \hat{\sigma}, \boldsymbol{pk}, (Y, y)). \end{array}\right] = 1.$$

Definition 4 (Witness Extractability): Assuming the following condition holds for each $\mathcal{PPT}$ *adversary* $\mathcal{A}$ *executing the experiment* $aWitExt_{\mathcal{A},\prod_{R_e,\Sigma}}$ *detailed in Algorithm 2, then a linkable ring adaptor signature scheme* $\prod_{R_e,\Sigma}$ *satisfies witness extractability:*

$$\Pr[\text{aWitExt}_{\mathcal{A},\prod_{R_e,\Sigma}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Definition 5 (aEUF–CMA Security): Assuming the following condition holds for each $\mathcal{PPT}$ *adversary* $\mathcal{A}$ *executing the experiment* $aSignForge_{\mathcal{A},\prod_{R_e,\Sigma}}$ *detailed in Algorithm 3, then a linkable ring adaptor signature scheme* $\prod_{R_e,\Sigma}$ *is provably secure in the aEUF-CMA security model:*

$$\Pr[\text{aSignForge}_{\mathcal{A},\prod_{R_e,\Sigma}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Definition 6: (Pre-signature Anonymity): *Assuming the following condition holds for each* $\mathcal{PPT}$ *adversary* $\mathcal{A}$ *executing the experiment* $aAnon_{\mathcal{A},\prod_{R_e,\Sigma}}$ *detailed in Algorithm 4, then a linkable ring adaptor signature scheme* $\prod_{R_e,\Sigma}$ *achieves pre-signature anonymity:*

$$\left|\Pr[\text{aAnon}_{\mathcal{A},\prod_{R_e,\Sigma}}(\lambda) = 1] - \frac{1}{2}\right| \leq \text{negl}(\lambda).$$

Algorithm 3 Experiment $\mathbf{aSignForge}_{\mathcal{A},\prod_{\mathbf{Re},\Sigma}}$

Procedure: $\mathbf{aSignForge}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda)$**:**
- $\Omega=\emptyset, \mathcal{F}=\emptyset$; $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in 1, n$];
- $\boldsymbol{pk} = \{pk_1, pk_2, \cdots, pk_n\}$;
- $(M^*, pk_{i^*}, \hat{\boldsymbol{pk}}) \leftarrow$ $\mathcal{A}^{O_s, O_{ps}, O_{\text{Corrupt}}}(\boldsymbol{pk})$, where $pk_{i^*} \in \hat{\boldsymbol{pk}} \subseteq \boldsymbol{pk}$;
- $(Y, y) \leftarrow \text{LockGen}(\lambda)$; $\hat{\sigma} \leftarrow \text{PreSign}(sk_{i^*}, \hat{\boldsymbol{pk}}, Y, M^*)$;
- $(\tilde{\boldsymbol{pk}}, \sigma^*) \leftarrow \mathcal{A}^{O_s, O_{ps}, O_{\text{Corrupt}}}(\hat{\sigma}, Y)$;
- return $((\tilde{\boldsymbol{pk}} \subseteq \boldsymbol{pk} \backslash \mathcal{F}) \wedge ((\star, M^*, \tilde{\boldsymbol{pk}}) \notin \Omega) \wedge \text{Verify}(\tilde{\boldsymbol{pk}}, \sigma^*, M^*) = 1)$;
- **Procedure:** $O_{Corrupt(i)}$*:*
  - $\mathcal{F} = \mathcal{F} \cup \{pk_i\}$;
  - return sk_i;
- **Procedure:** $O_s(M, i, \bar{\boldsymbol{pk}})$*:*
  - $\sigma \leftarrow \text{Sign}(sk_i, M, \bar{\boldsymbol{pk}})$, where $pk_i \in \bar{\boldsymbol{pk}} \subseteq \boldsymbol{pk}$;
  - $\Omega = \Omega \cup \{i, M, \bar{\boldsymbol{pk}}\}$;
  - return σ;
- **Procedure:** $O_{ps}(M, i, \bar{\boldsymbol{pk}}, Y)$*:*
  - $\hat{\sigma} \leftarrow \text{PreSign}(\bar{\boldsymbol{pk}}, sk_i, Y, M)$, where $pk_i \in \bar{\boldsymbol{pk}}$ and $Y \in L_R$;
  - $\Omega = \Omega \cup \{i, M, \bar{\boldsymbol{pk}}\}$;
  - return $\hat{\sigma}$;

Algorithm 4 Experiment $\mathbf{aAnon}_{\mathcal{A},\prod_{\mathbf{Re},\Sigma}}$

Procedure: $\mathbf{aAnon}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda)$**:**
- $\Omega=\emptyset$; $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in [1, n]$;
- $\boldsymbol{pk} = \{pk_1, pk_2, \cdots, pk_n\}$;
- $(M^*, i_0, i_1, \hat{\boldsymbol{pk}}, Y) \leftarrow$ $\mathcal{A}_1{}^{O_s, O_{ps}}(\boldsymbol{pk})$, where $pk_{i_o}, pk_{i_1} \in \hat{\boldsymbol{pk}} \subseteq \boldsymbol{pk}$; /* O_s, O_{ps} same as Algorithm 3 */
- $b \xleftarrow{\$} \{0, 1\}$; $\hat{\sigma} \leftarrow \text{PreSign}(\hat{\boldsymbol{pk}}, sk_{i_b}, Y, M^*)$;
- $b' \leftarrow \mathcal{A}_2{}^{O_s, O_{ps}}(\hat{\sigma}, Y)$;
- return $(b = b' \wedge \{i_0, \star, \star\} \notin \Omega \wedge \{i_1, \star, \star\} \notin \Omega)$;

Definition 7 (Linkability): Assuming the following condition holds for each PPT adversary $\mathcal{A}$ executing the experiment $aLink_{\mathcal{A},\prod_{Re,\Sigma}}$ detailed in Algorithm 5, then a linkable ring adaptor signature scheme $\prod_{R_e,\Sigma}$ achieves Linkability:

$$\Pr[\text{aLink}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Definition 8 (Non-Frameability): Assuming the following condition for each PPT adversary $\mathcal{A}$ executing the experiment $aNonFra_{\mathcal{A},\prod_{Re,\Sigma}}$ detailed in Algorithm 6, then a linkable ring adaptor signature scheme $\prod_{R_e,\Sigma}$ achieves Non-frameability:

$$\Pr[\text{aNonFra}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

C. Security and Privacy Definition of PCN

We leverage Canetti's *Universal Composability* (UC) framework [22] to describe security and privacy in the context of concurrent executions. Thus, while maintaining security and privacy requirements, we enable the composition of linkable dualring adaptor signatures with other application-dependent protocols. Due to space limitations, detailed descriptions of these protocols will be provided in Appendix A-A in the Supplementary Material.

Algorithm 5 Experiment $\mathbf{aLink}_{\mathcal{A},\prod_{\mathbf{Re},\Sigma}}$

Procedure: $\mathbf{aLink}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda)$**:**
- $\Omega=\emptyset, \mathcal{F}=\emptyset$; $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in [1, n]$;
- $\boldsymbol{pk} = \{pk_1, pk_2, \cdots, pk_n\}$;
- $(\hat{\boldsymbol{pk}}_i, M_i^*, \sigma_i^*)_{i=1,2} \leftarrow \mathcal{A}^{O_{\text{Corrupt}}, O_{ps}, O_s}(\boldsymbol{pk})$;
- /* O_{Corrupt}, O_{ps}, O_s same as Algorithm 3 */
- $b_1 = \{\star, M_i^*, \hat{\boldsymbol{pk}}_i\}_{i=1,2} \notin \Omega$;
- $b_2 = \text{Verify}(M_0^*, \hat{\boldsymbol{pk}}_0, \sigma_0^*) \wedge (\text{Verify}(M_1^*, \hat{\boldsymbol{pk}}_1, \sigma_1^*)$;
- $b_3 = \text{Link}((M_0^*, \hat{\boldsymbol{pk}}_0, \sigma_0^*), (M_1^*, \hat{\boldsymbol{pk}}_1, \sigma_1^*))$;
- $b_4 = \left|((\hat{\boldsymbol{pk}}_0 \cup \hat{\boldsymbol{pk}}_1) \cap \mathcal{F}) \cup ((\hat{\boldsymbol{pk}}_0 \cup \hat{\boldsymbol{pk}}_1) \backslash \boldsymbol{pk})\right|$;
- return $(b_1 = 1 \wedge b_2 = 1 \wedge b_3 = \textbf{Unlinked} \wedge b_4 \leq 1)$;

Algorithm 6 Experiment $\mathbf{aNonFra}_{\mathcal{A},\prod_{\mathbf{Re},\Sigma}}$

Procedure: $\mathbf{aNonFra}_{\mathcal{A},\prod_{Re,\Sigma}}(\lambda)$**:**
- $\Omega=\emptyset, \mathcal{F}=\emptyset$; $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in [1, n]$;
- $\boldsymbol{pk} = \{pk_1, pk_2, \cdots, pk_n\}$;
- $(M, \hat{\boldsymbol{pk}}, \sigma) \leftarrow \mathcal{A}^{O_{\text{Corrupt}}, O_{ps}, O_s}(\boldsymbol{pk})$;
- /* O_{Corrupt}, O_{ps}, O_s same as Algorithm 3 */
- $b_1 = \text{Verify}(M, \hat{\boldsymbol{pk}}, \sigma)$; $b_2 = (\{\star, \star, \sigma\} \notin \Omega)$;
- $b_3 = (\exists\{M', \hat{\boldsymbol{pk}}', \sigma'\} \in \Omega$ $s.t.$ $\text{Link}((M, \hat{\boldsymbol{pk}}, \sigma), (M', \hat{\boldsymbol{pk}}', \sigma')) = \textbf{Linked})$;
- $b_4 = ((\boldsymbol{pk} \cap \hat{\boldsymbol{pk}}' \cap (\hat{\boldsymbol{pk}} \backslash \mathcal{F})) \neq \emptyset)$;
- return $(b_1 = 1 \wedge b_2 = 1 \wedge b_3 = 1 \wedge b_4 = 1)$;

1) Universal Composability: When a user running protocol τ interacts with attacker $\mathcal{A}$, the output set of environment ε through all relevant machines for random coins is declared as $\text{EXEC}_{\tau,\mathcal{A},\varepsilon}$.

Definition 9: (Universal Composability): *Assuming for each PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for any environment ε, the ensembles $EXEC\tau, \mathcal{A}, \varepsilon$ and $EXEC_{\mathcal{F},\mathcal{S},\varepsilon}$ are computationally indistinguishable, then we state that a protocol τ UC-realizes an ideal functionality $\mathcal{F}$.*

2) Ideal Functionality: We provide a formal definition of the ideal-world function $\mathcal{F}$ for the anonymity enhanced locking mechanism. To ensure modularity, we adopt a similar approach to [3], modeling the functionality of the dongle using the UC definition instead of a complete PCN formulation. To maintain simplicity, we make the assumption that each user pair only creates one link in each direction. However, it is worth noting that the model can be readily extended to accommodate more general cases if needed. Additional assumptions and details of the ideal functionality interface are provided in Appendix A-A in the Supplementary Material.

In what follows we delve into how ideal functionality $\mathcal{F}$ capture AEMHL-related security and privacy concepts.

3) Atomicity: In simpler terms, atomicity ensures that each node in a payment path can release their left lock if the right lock has already been released. This is achieved through the functionality $\mathcal{F}$, which keeps track of the locks' status in the list $\mathcal{L}$ and provides the Release interface to release a lock (lid) if it is locked and the subsequent lock (getNextLock(lid)) has been released.

4) Consistency: AEMHL achieves consistency by ensuring that no adversary can release their left lock before the right lock is released first. This prevents situations where an AEMHL is prematurely released before reaching the receiver. The ideal functionality models this property by allowing a node to release the left lock only if the right lock has already been released or if the user is the receiver.

5) Anonymity-Enhancement: It includes two angles of relational anonymity and identity anonymity. Among them, each intermediary node must not discover any secret message regarding the group of nodes in an AEMHL beyond his neighbors to maintain relationship anonymity [23]. The lock identification are sampled at random, and throughout the locking phase, a node only learns the identification of his left and right lock as well as his left and right direct neighbor, so $\mathcal{F}$ satisfies this requirement. Identity anonymity further requires that each node does not learn the original public key of any lock other than his neighbors.

IV. Anonymity-Enhancing Multi-Hop Locks for Monero PCN

In this section, we will present the formal definition of AEMHL and provide security proof for its security properties. Before that, we refer readers to Appendix A-B in the Supplementary Material for an introduction to the cryptographic building blocks combined in the system construction.

A. Construction of AEMHL

To overcome the lack of anonymity and Linkability associated with existing LRAS signature schemes, and to align the foundational cornerstone with our goal of PCN anonymity and efficiency, we propose a novel general structure for LRAS signatures (shown in the algorithm 7) that allow any homomorphic one-way function. Common examples of homomorphic one-way functions include the learning with errors problem [24] and discrete logarithm.

It is worth noting that in comparison to the counterpart presented in [18], our LRAS employs standard loop calculations of ring signatures(such as $j+1$ to $j-1$), during both the pre-signature and pre-verification stages. This approach virtually enables the formation of a ring signature structure. Besides the parameter L, which is introduced to accommodate the adapter signature, the ring structure in our LRAS scheme aligns closely with the signature structure of the underlying Monero blockchain. Additionally, to facilitate the adaptation of algorithms with different algebraic structures to this solution, we introduced a function $V_A(\cdot,\cdot)$ for the computation of parameter L in various scenarios(e.g., the generator h can be used as a base or exponent).

In this paper, the ideal functionality for a channel with anonymous communication is set to $\mathcal{F}_{\text{anon}}$, the homomorphic one-way function is set to $A(\cdot):\mathbb{R}\rightarrow\mathbb{D}$, and the homomorphic one-way function with different generators is set to $V_A(\cdot,\cdot)$. Furthermore, we present successive $\odot$ and $\otimes$ operations, respectively, using the symbols $\bigodot$ and $\bigotimes$:

$$\bigodot_{i=1}^{n} A_i = A_1 \odot A_2 \cdots A_{n-1} \odot A_n,$$

$$\bigotimes_{i=1}^{n} b_i = b_1 \otimes b_2 \otimes \cdots \otimes b_{n-1} \otimes b_n.$$

Our generic construction and its Schnorr-based instantiation algorithms are shown in Figure 1 and Appendix B-A in the Supplementary Material, respectively.

1) KeyGen Phase: Calls to $\mathcal{F}_{\text{kgen}}$ form the body of the Generation algorithm. After this stage, user U_i and user U_j will obtain their public-private key pair (x_i, pk) and (x_j, pk) respectively, so that $pk = A(x_i \otimes x_j)$.

2) Setup Phase: The user U_0 initializes the path of channel by sampling n values $(y_0, \cdots, y_{n-1})$ from the domain of $\mathbb{R}$. Then it proves that U_0 holds $y = \bigotimes_{j=0}^{i} y_j$ s.t. $Y_i = A(y)$ by NIZK algorithm, convincing U_i that U_0 is credible and the triple it received is valid. Finally, it sends the first key k_n to U_n via $\mathcal{F}_{\text{anon}}$, and sends each intermediate node a triple $(A(\bigotimes_{j=0}^{i-1} y_j), A(\bigotimes_{j=0}^{i} y_j), (\pi_i, y_i))$ respectively. Then the triplet will be verified by the intermediate node U_i using the homomorphic attribute of $A(\cdot)$ to verify whether it is correctly generated.

3) Locking Phase: Before the Locking phase, the statement Y_i of the witness and the transaction tx to be signed are privately negotiated and finalized by users U_i and U_{i+1}. The main aim of the Locking algorithm is to calculate the partial signature of the transaction tx in a distributed manner between the two parties. First, the two parties confirm their identities by verifying the common public key and determine n groups of values $z_i \in \Delta_z (i \in [0, n-1])$ randomly selected and the common index j. Based on this, both parties can hide their real identities in n groups of public keys pk_i using index j. Then, they compute respectively R_0, R_1 using selected ephemeral key r_0, r_1. Simultaneously with the first round of interactions, they exchange the value R through the commitment algorithm and verify its validity through NIZK. Then $R_0 \odot R_1$ as a random value commitment of the signature is calculated by both parties involved and participates in subsequent calculations. But because they don't learn the discrete logarithm of Y_i, the signature still cannot be completed at this time. Under normal circumstances, $z_j = Z(sk_i \otimes sk_{i+1}, r_0 \otimes r_1, 2c_j)$ will be jointly calculated by both parties, but this value lacks Y_i in the random value commitment. Actually, the calculated z_j also cannot be used as a usable signature on the message tx about transaction due to the lack of the additive item y^* (where $A(y^*) = Y_i = Y_{i-1} \odot A(y_i)$). Eventually, both parties will sequentially merge z_j into the random value set $\vec{z}$ and save these tuples. Among them, $(R_0 \odot R_1 || L_0 \odot L_1, \vec{z})$ is used as the left identification s_{i+1}^L of U_{i+1}, and the set of $\vec{z}$ is used as the right identification s_i^R of U_i.

4) Release Phase: U_{i+1} first parses his own identity s_{i+1}^L, s_{i+1}^I and the received unlocking key k_{i+1} into $(W_0, \boldsymbol{w_1})$, (Y_i', Y_i, y_i) and $(R_{i+1}, \boldsymbol{s_{i+1}})$. Then U_{i+1} respectively extracts the locked value both of the current channel and the specific position of the next channel (where it is assumed that the real locked value position of the latter channel is v) and updating the locked value $z_j' = z_j \otimes s_v \oslash \otimes(s_R \otimes y \otimes y)$. Finally, U_{i+1} puts z_j' in the set $\boldsymbol{w_0}$ and returns $(R_i, \boldsymbol{s_i})$ as the unlock key k_i of the previous channel. This means that the effective signature on the transaction can be calculated by U_{i+1} after the discrete logarithm of the statement Y_i of the witness is revealed. Knowing this, the open with atomicity is enforced by U_{i+1}, and the lock between U_{i+1} and U_{i+2} bounded by $Y_{i+1} = Y_i \odot A(y_{i+1})$ is opened. In this perspective, the value $y^* \otimes y_{i+1}$ is revealed after opening the right lock, and user U_{i+1} can immediately use his secret value y_{i+1} to extract y^* and release its left lock with a usable formal signature on the transaction.

5) Verify Phase: This step is mainly used by the channel to verify the validity of the generated unlock key. The channel user U_i first parses the lock information l_i and unlock the key value k_i, then calculate R_i, L_i and c_{i+1} for i from 1 to n loop

Algorithm 7 Generic Construction of Linkable Ring Adaptor Signature

Procedure Setup(λ) **:**
  define $H : \{0,1\}^* \to \Delta_c, H' : \{0,1\}^* \to \mathbb{G}$;
  return $param \leftarrow \text{TA.Setup}(\lambda)$;

Procedure KeyGen$(param)$ **:**
  return $(pk, sk) \leftarrow \text{TA.KeyGen}(param)$;

Procedure PreSign$(param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, sk_j, Y)$ **:**
  $h = H'(\boldsymbol{pk}), I = V_A(h, sk_j), r \leftarrow_\$ \Delta_r, z_i \leftarrow_\$ \Delta_c$ for all $i \neq j$;
  $R_j = A(r) \odot Y$; $L_j = V_A(h, r)$;
  $c_{j+1} = H(M, \boldsymbol{pk}, R_j, L_j, I)$;
  For $i = j+1$ *to* $j-1$**:**
    $R_i = A(z_i) \odot Y \odot V'(pk_i, c_i)$;
    $L_i = V_A(h, z_i) \odot V'(I, c_i)$;
    $c_{i+1} = H(M, \boldsymbol{pk}, R_i, L_i, I)$;
  $\hat{z_j} = Z(sk_j, r, c_j)$;
  return $\hat{\sigma} = (c_1, \hat{\boldsymbol{z}}, I)$ where $\hat{\boldsymbol{z}} = (z_1, \cdots, \hat{z_j}, \cdots z_n)$;

Procedure PreVerify$(param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma}, Y)$ **:**
  parse $\hat{\sigma} = (c_1, \hat{\boldsymbol{z}}, I)$; $h = H'(\boldsymbol{pk})$;
  For $i = 1$ *to* $n-1$**:**
    $R_i = A(z_i) \odot Y \odot V'(pk_i, c_i)$;
    $L_i = V_A(h, z_i) \odot V'(I, c_i)$;
    $c_{i+1} = H(M, \boldsymbol{pk}, R_i, L_i, I)$;
  $R_n = A(z_n) \odot Y \odot V'(pk_n, c_n)$;
  $L_n = V_A(h, z_n) \odot V'(I, c_n)$;
  if $c_1 \neq H(M, \boldsymbol{pk}, R_n, L_n, I)$ then return 0;
  return 1;

Procedure Adapt$((Y, y), \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma}, M)$ **:**
  parse $\hat{\sigma} = (c_1, \hat{\boldsymbol{z}}, I)$; $z_j = \hat{z_j} \otimes y$;
  return $\sigma = (c_1, \boldsymbol{z}, I)$ where $\boldsymbol{z} = (z_1, \cdots, z_j, \cdots z_n)$;

Procedure Verify$(param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \sigma)$ **:**
  parse $\sigma = (c_1, \boldsymbol{z}, I)$; $h = H'(\boldsymbol{pk})$;
  For $i = 1$ *to* $n-1$**:**
    $R_i = A(z_i) \odot V'(pk_i, c_i)$;
    $L_i = V_A(h, z_i) \odot V'(I, c_i)$;
    $c_{i+1} = H(M, \boldsymbol{pk}, R_i, L_i, I)$;
  $R_n = A(z_n) \odot V'(pk_n, c_n)$; $L_n = V_A(h, z_n) \odot V'(I, c_n)$;
  if $c_1 \neq H(M, \boldsymbol{pk}, R_n, L_n, I)$ then return 0;
  return 1;

Procedure Ext$(Y, \hat{\sigma}, \sigma)$ **:**
  parse $\hat{\sigma} = (c_1, \hat{\boldsymbol{z}}, I)$ *and* $\sigma = (c_1, \boldsymbol{z}, I)$; $y = z_j \oslash \hat{z_j}$;
  if $Y = A(y)$ then return y;
  return $\perp$;

Procedure Link$(\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \sigma', \sigma'')$ **:**
  parse $\sigma' = (c_1', \boldsymbol{z}', I')$ *and* $\sigma'' = (c_1'', \boldsymbol{z}'', I'')$;
  if $I' = I''$ then return **Linked**;
  return **Unlinked**;

(where $i = n$, no longer calculates the last c_{i+1}). Finally, the verification passes if $c_1 = H(\boldsymbol{pk}||R_n||L_n||tx)$; otherwise, the verification fails.

B. Security Analysis

In the next sections that follow, we will briefly review our ideal functionality and will present the security analysis of our construction in more detail.

1) Key Generation Functionalities: Figure 2 presents the ideal functionality $\mathcal{F}_{\text{kgen}}$ used in the KeyGen stage of AEMHL. It simulates the key generation of a two-party Type-T signature, an issue that has been extensively researched (e.g., [25]).

Theorem 1: The protocal in Figure 1 UC-realizes the ideal functionality $\mathcal{F}$ in the $(\mathcal{F}_{\text{kgen}}, \mathcal{F}_{\text{syn}}, \mathcal{F}_{\text{smt}}, \mathcal{F}_{\text{anon}})$-hybrid model if COM is a robust commitment scheme, NIZK is a non-interactive zero-knowledge proof and $A(\cdot)$ is a homomorphic one-way function. (please see [22], [26], [27] for the concrete functionality or examples of F_{syn}, F_{smt}, and F_{anon} respectively and see A in the Supplementary Material for the relationships between them and AEMHL)

Proof: In the subsequent hybrids we define, the initial experiment is gradually changed.

$\mathcal{H}_0$: is the same as the procedure outlined in Figure 1.

$\mathcal{H}_1$: Interactions in this hybrid with the ideal functionality $\mathcal{F}_{\text{com}}$ showed in Figure 3 replace all uses of the commitment scheme compared to $\mathcal{H}_0$.

Rather than directly invoking the Commit algorithm with a certain message M to be signed, the parties now transmit a message in the form of Commit(sid, M) to the ideal functionality. Similarly, the Decommit algorithm is substituted with a invocation to algorithm Decommit(sid). As for the verifying party, it merely store the messages received from $\mathcal{F}_{\text{com}}$.

$\mathcal{H}_2$: Interactions in this hybrid with the ideal functionality $\mathcal{F}_{\text{NIZK}}$ described in Figure 4 replace all uses of the NIZK scheme compared to $\mathcal{H}_1$.

$\mathcal{H}_3$: In this hybrid, the tuple (l_i, l_{i+1}, s^L, s^R), the public-private key pair (sk_i, pk), the honest node U_i, the state s^I and other variable collections interact with $\mathcal{A}$ such that

$$\{\cdot, (l_i, s^L)\} \leftarrow \left\langle \cdot, \text{Lock}_{U_i}(s^I, sk_i, pk) \right\rangle$$

and

$$\{(l_{i+1}, s^R), \cdot\} \leftarrow \left\langle \text{Lock}_{U_i}(s^I, sk_i, pk), \cdot \right\rangle$$

The experiment ends if the adversary returns some k for all collection of these variables such that $\text{Vf}(l_{i+1}, k) = 1$ and $\text{Vf}(l_i, \text{Rel}(k, (s^I, s^L, s^R))) \neq 1$.

$\mathcal{H}_4$: In this hybrid, a public-private key pair (sk_i, pk), a collection of possibly corrupted nodes $(U_1, \cdots, U_n)$, a pair of honest nodes (U_0, U_i), a collection of initial states

$$(s_0^I, \cdots, s_n^I) \leftarrow \left\langle \begin{matrix} \text{Setup}_{U_0}(1^\lambda, U_1, \cdots, U_n), \\ \cdots, \\ \text{Setup}_{U_n}(1^\lambda) \end{matrix} \right\rangle,$$

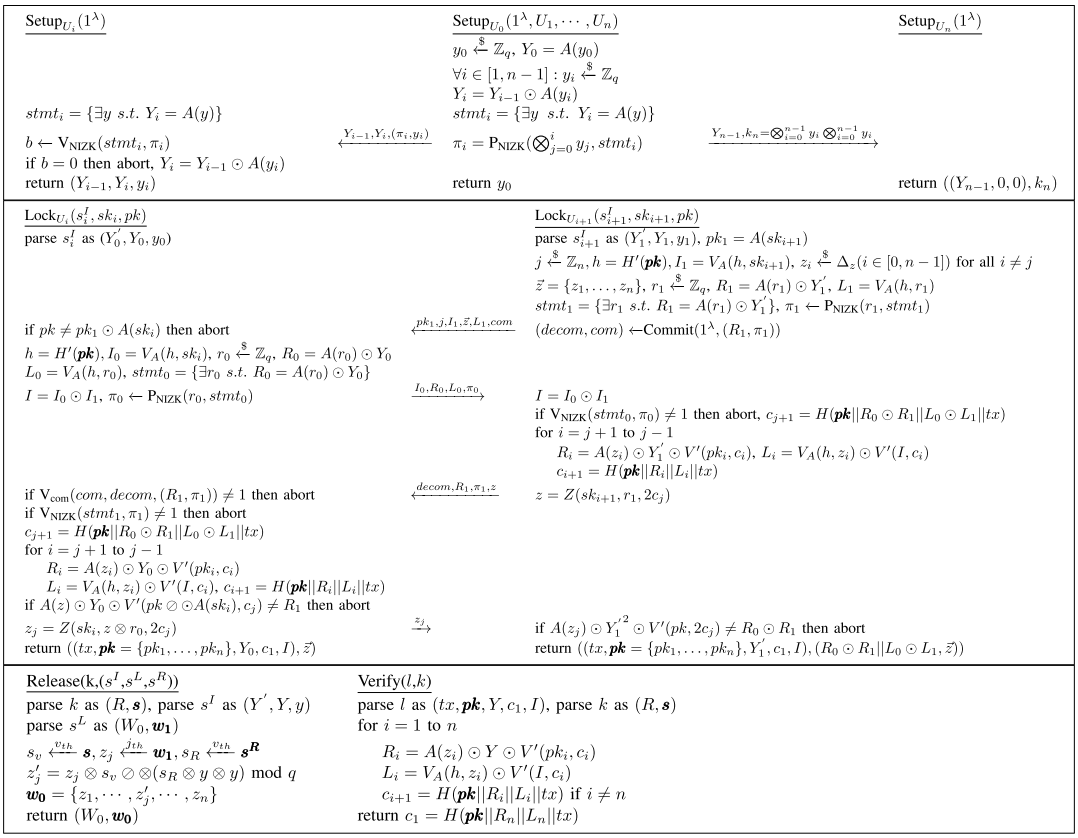

Fig. 1. Generic construction of AEMHL.

KeyGen$(param)$
Upon invocation by both U_0 and U_1 on input $(param)$:
select a hash function $H : \{0,1\}^* \to \mathbb{R}$
sample $x \leftarrow \mathbb{R}$ and compute $pk = A(x)$
set $sk_{U_0,U_1} = x$
sample x_0 and x_1 randomly s.t. $x = x_0 \otimes x_1$
send (x_0, pk, H) to U_0 and (x_1, pk, H) to U_1
ignore future calls by (U_0, U_1)

Fig. 2. Ideal functionality $\mathcal{F}_{\mathrm{kgen}}$ for key generation.

Commit(sid, M)
Upon execution by U_i (where $i \in \{0,1\}$):
store (sid, i, M) and return (com, sid) to U_{1-i}
if some $(sid, \cdot, \cdot)$ is already recorded, then disregard the message
Decommit(sid)
Upon execution by U_i (where $i \in \{0,1\}$):
if (sid, i, M) is stored, then return $(decom, sid, M)$ to U_{1-i}

Fig. 3. Ideal functionality $\mathcal{F}_{\mathrm{com}}$.

and a pair of locks (l_{i-1}, l_i) interact with $\mathcal{A}$ such that

$$\{\cdot, (l_{i-1}, \cdot)\} \leftarrow \left\langle \cdot, \mathrm{Lock}_{U_i}(s_i^I, sk_i, pk) \right\rangle$$

and

$$\{(l_i, \cdot), \cdot\} \leftarrow \left\langle \mathrm{Lock}_{U_i}(s_i^I, sk_i, pk), \cdot \right\rangle.$$

The experiment ends if the adversary returns some k_{i-1} for all collection of these variables such that $\mathrm{Vf}(l_{i-1}, k_{i-1}) = 1$ before the node U_i outputs a key k_i such that $\mathrm{Vf}(l_i, k_i) = 1$.

$\mathcal{H}_5$: Let $S = (U_0, \cdots, U_m)$ be an ordered set of nodes that may have corrupted users. Assuming node U_i is honest and $(U_{i+1}, \cdots, U_j)$ has been corrupted, we state that the ordered subset $S_A = (U_i, \cdots, U_j)$ to be adversarial. The tandem of adversarial subsets $S = (S_{A_1} || \cdots || S_{A_{1_{m'}}})$ is used to represent each group of node sets. It initializes a separate lock for each subset $(S_{A_i}, S^0_{A_{i+1}})$ when an honest node is asked to establish a lock for a collection $S = (S_{A_1} || \cdots || S_{A_{m'}})$. If existing, $S^0_{A_{i+1}}$ is the first item of the $(i+1)$-th collection. It will release the key for the new lock $(S_{A_1} || \cdots || S_{A_{m'}})$ when an honest user $S^0_{A_{i+1}}$ is asked to turn over the key of the associated lock. At this time, it means that each adversarial subset is statically corrupted by only one different adversary, and the shared public key pk_{lock} of each lock is hidden, which satisfies relational anonymity and strong identity anonymity between each subset.

Prove(sid, x, w)
Upon execution by U_i (for $i \in \{0,1\}$):
if $R_e(x, w) = 1$, then return $(proof, sid, x)$ to U_{1-i}

Fig. 4. Ideal functionality $\mathcal{F}_{\mathrm{NIZK}}$.

$\mathcal{S}$: The only difference between the simulator's interaction and that of $\mathcal{H}_3$ is that simulator's behaviors are determined by its interaction with ideal functionality $\mathcal{F}$. The $\mathcal{S}$ is asked by adversary $\mathcal{A}$ on the following collection of inputs after reading the communications between adversary $\mathcal{A}$ and the honest nodes through $\mathcal{F}_{\mathrm{anon}}$.

1) $(\cdot, \cdot, \cdot, \cdot,$ Init): The $\mathcal{S}$ creates a new lock chain and rebuilds the adversarial collection using the ids.

2) $(\cdot,$ Lock): Initiating the locking process with the $\mathcal{A}$, the $\mathcal{S}$ responds with $\perp$ if the execution is unsuccessful.

3) $(\cdot,$ Rel) The $\mathcal{S}$ publishes and releases the key for the relevant lock.

The simulator requests the matching interface of $\mathcal{F}$ if $\mathcal{A}$ interacts with an honest node.

Keep in mind that the $\mathcal{S}$ works well and interacts with the ideal world as the $\mathcal{F}$. Additionally, the simulation always reflects the ideal world. That means it will stop if the actions of $\mathcal{F}$ are not accommodated by the interfaces of ideal functionality. What needs to be demonstrated is that the adjacent hybrids are indistinguishable from the sight of environment ε. □

PreSign(M, y)
Upon invocation by both U_0 and U_1 on input (M, y):
compute $(R, z_j) = \text{Sig}_{\text{LRAS}}(sk_{U_0,U_1}, M, A(y))$
return $(R, \mathbf{z} = \{z_1, \cdots, z_j, \cdots, z_n\})$

Fig. 5. Ideal functionality $\mathcal{F}_{\text{PreSign}}$ for signing.

Lemma 1: For all $\mathcal{PPT}$ distinguishers ε, it holds that

$$\text{EXEC}_{\mathcal{H}_0,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{H}_1,\mathcal{A},\varepsilon}.$$

Proof: Its security directly results from the security of the commitments COM. □

Lemma 2: For all $\mathcal{PPT}$ distinguishers ε, it holds that

$$\text{EXEC}_{\mathcal{H}_1,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{H}_2,\mathcal{A},\varepsilon}.$$

Proof: Its security directly results from the security of the non-interactive zero-knowledge scheme NIZK. □

Lemma 3: For all $\mathcal{PPT}$ distinguishers ε, it holds that

$$\text{EXEC}_{\mathcal{H}_2,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{H}_3,\mathcal{A},\varepsilon}.$$

Proof: First of all, we hand over the locking algorithm process to the ideal functionality as shown in Figure 5, where sk_{U_0,U_1} is established by $\mathcal{F}_{\text{kgen}}$ proposed earlier.

The upper limit of the number of interactions is set to q in this paper. we will prove $Pr[abort|\mathcal{H}_3] \leq negl(\lambda)$ below to show that $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable. Now assuming that the above negative proposition is true, we can construct a reduction algorithm to break the EUF of LRAS:

The random index $j \in [1, q]$ and public key pk are given to the reduction algorithm. Every request made to the signature algorithm converts to a question to the signature oracle. Let X be the public key of the U_j for key generation. The algorithm sets $X = pk$ when the adversary interacts with the j-th user. If an event triggering $\mathcal{H}_3$ abort occurs, the reduction algorithm return $(k^*, l^*) = ((R, \mathbf{s}), (M^*, pk^*))$ using signature oracle. Otherwise the experiment aborts.

Now amusing that the sequence number of the interaction that triggers the $\mathcal{H}_3$ abort is j, and the sequence number of the current lock l^* is $i + 1$. The left identification s_i^L and key k^* will be parsed into $(W_{i,0}, \mathbf{w}_{i,1})$ and $(R^*, \mathbf{s}^*)$ by the release algorithm when the abort incident is triggered. And then the release algorithm will return $(W_{i,0}, \mathbf{w}_{i,0})$, where $(z_j \otimes s_v \oslash \otimes(s_R \otimes y \otimes y)) \subset \mathbf{w}_{i,0}$. So we can do the conversion as follows:

$$\begin{aligned} & z_j \otimes s_v \oslash \otimes(s_R \otimes y \otimes y) \\ &= s_i \oslash \bigotimes_{j=0}^{i-1}(y_j) \oslash \bigotimes_{j=0}^{i-1}(y_j) \otimes s_v \oslash \otimes \\ & \quad s_j \oslash \bigotimes_{j=0}^{i}(y_j) \oslash \bigotimes_{j=0}^{i}(y_j) \otimes y \otimes y \\ &= s_i \otimes s_v \oslash \otimes s_j \end{aligned}$$

where s_j is the response obtained from asking the random oracle for m_j in the j-th session. This implies that there is $s^* \neq s_j$. Otherwise, the valid signature generated by the random oracle is returned to the adversary. Since the messages of each session are different, $((R, \mathbf{s}), (M^*, pk^*))$ is a pair of valid forgery. In summary, there is at least a $\frac{1}{q \cdot poly(\lambda)}$ probability that an $\mathcal{A}$ will succeed in breaking the EUF of the LRAS. Furthermore, regardless of whether the adversary corrupts the left or right node of the channel, the simulator responds appropriately and checks whether the obtained witness is valid. □

Lemma 4: For all $\mathcal{PPT}$ distinguishers ε, it holds that

$$\text{EXEC}_{\mathcal{H}_3,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{H}_4,\mathcal{A},\varepsilon}.$$

Proof: Let q denote the bound on the number of interactions. we will prove $Pr[abort|\mathcal{H}_4] \leq negl(\lambda)$ below to show that $\mathcal{H}_3$ and $\mathcal{H}_4$ are indistinguishable. Now assuming that the above negative proposition is true, we can construct a reduction algorithm to break the EUF of LRAS: Set the commitment value Y^* corresponding to the user U_i, and the session index is j. When the i-th node is asked in the setup phase, returns $Y_i = Y^*$; when index $i \in [0, i-1]$ of users are asked, returns $(Y_i = Y_{i+1} \otimes A(y_{i+1} \otimes y_{i+1})^{-1}, Y_{i+1}, y_i)$; when index $i \in [i+1, n-1]$ of users are asked, return $(Y_i = Y_{i-1} \otimes A(y_i \otimes y_i), Y_{i-1}, y_i)$. The reduction fails if user U_i is required to open the lock. Otherwise, the reduction algorithm parses the updated right identifier $\mathbf{s}^R$ from U_i and returns $s^* \otimes y_{i-1} \otimes y_{i-1} \otimes (s^R)^{-1} \in \boldsymbol{w_0}$ after the adversary finally outputs the unlock key k^*.

Since node U_i is honest and group $\mathbb{G}$ is an exchange group, from the perspective of $\mathcal{A}$, the updated setup algorithm's distribution is identical to the original distribution. The aborts of $\mathcal{H}_4$ will only happen when the output k^* is valid and can successfully pass the release algorithm. We have $s_R \in \boldsymbol{s^R}$ in the form of $s' \otimes y^{-1} = Z(sk_i, Z(sk_{i+1}, r_1, 2c_j) \otimes r_0, 2c_j) \otimes y^{-1}$ for some y. Hence (R, s') can be determined to be a valid LRAS signature on the message M_{i-1} due to successful release. Therefore we have

$$\begin{aligned} & A(s* \otimes y_{i-1} \otimes y_{i-1} \otimes (s^R)^{-1}) \\ &= A(s* \otimes y_{i-1} \otimes y_{i-1} \otimes (s')^{-1} \otimes y) \\ &= A(y_{i-1} \otimes y_{i-1} \otimes y \otimes y) \\ &= A(y_{i-1} \otimes y_{i-1}) \odot (Y^* \odot A(y_{i-1} \otimes y_{i-1})) \\ &= Y^*. \end{aligned}$$

In summary, the probability of $\mathcal{A}$ breaking the EUF of LRAS is at least $\frac{1}{q \cdot n \cdot poly(\lambda)}$. This in turn proves that the probability of $\mathcal{H}_3$ aborts is negligible(i.e. Lemma 4 holds). □

Lemma 5: For all $\mathcal{PPT}$ distinguishers ε it holds that

$$\text{EXEC}_{\mathcal{H}_4,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{H}_5,\mathcal{A},\varepsilon}.$$

Proof: According to the hybrid definition in $\mathcal{H}_5$ introduced earlier, different adversarial subsets are always linked by honest nodes. Therefore, there is always a witness y such that $Y_i = Y_{i-1} \otimes A(y)$. And the $\mathcal{A}$ at the end of the set is unknown y for each A_i in $\mathcal{H}_4$. This is the same as the adversary's perspective in $\mathcal{H}_5$ (i.e. the simulation does not aborts). □

Lemma 6: For all $\mathcal{PPT}$ distinguishers ε it holds that

$$\text{EXEC}_{\mathcal{H}_5,\mathcal{A},\varepsilon} \approx \text{EXEC}_{\mathcal{F},\mathcal{S},\varepsilon}.$$

Proof: Aside from possibly slightly different syntax and behavior, the execution of the environment is largely the same in both cases. □

V. Generic Construction of Linkable DualRing Adaptor Signatures

Building upon the structure of the LRAS, we have constructed a locking mechanism in the previous section that fulfills the initial privacy and simplicity requirements. But in order to further meet the practical needs for efficient and versatile deployment, we present two optimized schemes based on the LRAS that possess unique efficiency attributes.

To demonstrate the security of our signatures, we select the representative LDRAS for formal proof under the random oracle model.

A. Our New Basic Cryptographic Cornerstone

We give the generic construction of linkable dualring adaptor signature and its instantiations respectively in Algorithm 8 and Appendix B-B in the Supplementary Material. In addition, the locking mechanism based on the generic construction of the LDRAS has been placed in Appendix B-C in the Supplementary Material due to space reasons.

Given the intricate structure of LDRAS, we aim to provide a more comprehensive algorithmic description. Initially, the Setup and KeyGen phases initialize system parameters and generate the signer's public-private key pairs, respectively. Moving to the pre-signing stage, the signer first maps the public key set to the anonymous public key hash h of group $\mathbb{G}$, using this hash along with the private key to compute the link identification I. Subsequently, a random value r and $n-1$ random challenge values c_i are selected. These random values are then employed to calculate the commitment R that embeds statement Y, and the verification identifier L, used for auxiliary verification. The signer then substitutes R and L into the signature hash H to compute the total challenge value. Using the total challenge value and the chosen $n-1$ random challenge values, the signer calculates the actual challenge c_j. The partial signature z is then computed through the random value r, the private key sk, and the signer's actual challenge value. As the final signature takes the form $(z, c_1, \cdots, c_n, I)$, the signer's true challenge value is concealed within n challenge values. The pre-verification phase is conducted directly using the pre-signed content, anonymous public key collection, and statement, without requiring the signer's actual public key. The other stages mirror those of the adapter signature, including the pre-signing and pre-verification stages, with the exception of the generation of J in the adaptation and verification stages. The J value is primarily generated after the adapter successfully adapts the formal signature to pass verification. Additionally, the Link stage can identify whether two signatures were generated by the same signer by comparing the link identifiers of the two signatures.

Compared to the basic LRAS, the LDRAS scheme employs two independent rings and simpler group operations. This transforms the pre-signature and pre-verification processes from the original loop calculations to cumulative calculations, leading to a change in the composition structure of the signature. The new computing mode alters the signature from the original structure of a single challenge value and n response values to n challenge values and a single response value. This adjustment will facilitate the construction of shorter signatures in lattice-based encryption systems, effectively reducing storage overhead. In addition, by eliminating the constraints of loop calculations and performing commitment calculations only once, LDRAS simplifies the security proof, necessitating just one rewind simulation. In contrast, LRAS requires n rewinds due to the calculation of n commitments within the loop. This results in a more concise security reduction for LRAS.

1) Security Proof: We first add the challenge values c_i and accompanying decoy public keys pk_i to the commitment R using V'. The element $c_j = c \oslash \bigotimes_{i \neq j} c_i \in \Delta_c$ of challenge value collection is calculated by the signer indexed as j after receiving its actual challenge value c. The signer calculates z using the Type-T signature technique. All keys pk_i and their corresponding challenge value c_i are used to recreate the commitment R for verification. The value $\bigotimes_{i=1}^{n} c_i$ will match the actual challenge value c.

Theorem 2: The linkable dualring adaptor signature satisfies pre-signature adaptability.

Proof: It is known that $\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \mathbf{c} = \{c_1, \cdots, c_n\}$. We substitute $\hat{\sigma}$ into the PreVerify algorithm to calculate $R' = A(\hat{z}) \odot Y \odot \bigodot_{i=0}^{n-1} V'(pk_i, c_i)$, $L' = V_A(h, \hat{z}) \odot \bigodot_{i=0}^{n-1} V'(I, c_i)$, $c' = \bigotimes_{i=0}^{n-1} c_i$ and then calculate $\sigma = (z = \hat{z} \otimes y, \mathbf{c}, *)$ by Adapt algorithm. So we can calculate as follows:

$$
\begin{aligned}
c' &= H(M, \boldsymbol{pk}, R', L') \\
&= H(M, \boldsymbol{pk}, A(\hat{z}) \odot V'(\boldsymbol{pk}, \mathbf{c}) \odot Y, V_A(h, \hat{z}) \odot V'(I, \boldsymbol{c})) \\
&= H(M, \boldsymbol{pk}, A(\hat{z}) \odot Y \odot \bigodot_{i=0}^{n-1} V'(pk_i, c_i), V_A(h, \hat{z}) \\
&\quad \odot \bigodot_{i=0}^{n-1} V'(I, c_i)) \\
&= H(M, \boldsymbol{pk}, A(z) \odot A(y)^{-1} \odot Y \odot \bigodot_{i=0}^{n-1} V'(pk_i, c_i), \\
&\qquad V_A(h, r) \odot \bigodot_{i=0}^{n-1} V'(I, c_i) \oslash \odot V'(I, c_j)) \\
&= H(M, \boldsymbol{pk}, A(z) \odot \bigodot_{i=0}^{n-1} V'(pk_i, c_i), V_A(h, r) \\
&\quad \odot \bigodot_{i=0, i \neq j}^{n-1} V'(I, c_i)) \\
&= H(M, \boldsymbol{pk}, A(z) \odot V'(\boldsymbol{pk}, \mathbf{c}), L)
\end{aligned}
$$

Hence, Verify$(\boldsymbol{pk}, \sigma = (z, \mathbf{c}, *), M) = 1$ holds, that means σ is valid. □

Theorem 3: The linkable dualring adaptor signature satisfies witness extractability.

Proof: At present, a $\mathcal{PPT}$ adversary capable of breaking the witness extractability of the linkable dualring adapter signature is assumed to exist in our proof. And we state that all adversary queries to the random oracle can be simulated.

2) Simulate Phase: The simulator $\mathcal{S}$ sends a statement Y and a pre-signature $\hat{\sigma} = (\hat{z}, \mathbf{c}, I)$ on message M^* to adversary $\mathcal{A}$. These values holds relation $\hat{z} = Z(sk_j, r, c_j)$ and $c = H(\boldsymbol{pk}||R||L||M^*) = \bigotimes_{i=0}^{n-1} c_i$.

3) Challenge Phase: Finally, $\mathcal{A}$ outputs a usable formal $\sigma = (z^*, \mathbf{c}^*, I, J)$, where $R^* = A(z^*) \odot V'(\boldsymbol{pk}, \boldsymbol{c}^*)$ and $L^* = V_A(h, z^*) \odot J \odot V'(I, \boldsymbol{c}^*)$. It demonstrate that Ext $(Y, \sigma^*, \hat{\sigma})$ did not outputs $\perp$ if adversary win. Hence $\mathbf{c} = \mathbf{c}^*$. By the collision-resistant property of H, then $R = R^*, L = L^*$. It demonstrate $A(z^*) \odot V'(\boldsymbol{pk}, \boldsymbol{c}^*) = A(r) \odot Y \odot \bigodot_{i=1, i \neq j}^{n} V'(pk_i, c_i^*)$. The Ext algorithm allows us to calculate $y = z^* \oslash \hat{z}$. Thus, we have:

$$
\begin{aligned}
A(z^*) \odot V'(\boldsymbol{pk}, \boldsymbol{c}^*) &= A(\hat{z} \otimes y) \odot V'(\boldsymbol{pk}, \boldsymbol{c}^*) \\
&= A(\hat{z}) \odot Y \odot V'(\boldsymbol{pk}, \boldsymbol{c}) \\
&= V(\boldsymbol{pk}, \hat{z}, \mathbf{c}^*) \odot Y
\end{aligned}
$$

Algorithm 8 Generic Construction of Linkable DualRing Adaptor Signature

Procedure Setup(λ) :
  define $H : \{0,1\}^* \to \Delta_c$, $H' : \{0,1\}^* \to \mathbb{G}$;
  return $param \leftarrow$ TA.Setup(λ);
Procedure KeyGen($param$) :
  return $(pk, sk) \leftarrow$ TA.KeyGen($param$);
Procedure PreSign($param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, sk_j, Y$) :
  $h = H'(\boldsymbol{pk})$; $I = V_A(h, sk_j)$; $r \leftarrow_\$ \Delta_r$, $c_i \leftarrow_\$ \Delta_c$ for all $i \neq j$;
  $R = A(r) \odot Y \odot \bigodot_{i=1}^{n} V'(pk_i, c_i)$ for all $i \neq j$;
  $L = V_A(h, r) \odot \bigodot_{i=1}^{n} V'(I, c_i)$ for all $i \neq j$;
  $c = H(\boldsymbol{pk}||R||L||M)$; $c_j = c \oslash \bigotimes_{i=1}^{n} c_i$ for all $i \neq j$;
  $\hat{z} = Z(sk_j, r, c_j)$;
  return $\hat{\sigma} = (\hat{z}, \boldsymbol{c}, I)$ where $\boldsymbol{c} = (c_1, \cdots, c_n)$;
Procedure PreVerify($param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma} = (\hat{z}, \boldsymbol{c}, I), Y$) :
  $h = H'(\boldsymbol{pk})$; $R' = A(\hat{z}) \odot Y \odot \bigodot_{i=1}^{n} V'(pk_i, c_i)$;
  $L' = V_A(h, \hat{z}) \odot \bigodot_{i=1}^{n} V'(I, c_i)$; $c' = \bigotimes_{i=1}^{n} c_i$;
  return $c' = H(\boldsymbol{pk}||R'||L'||M)$;
Procedure Adapt($(Y, y), \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma}, M$) :
  parse $\hat{\sigma} = (\hat{z}, \boldsymbol{c}, I)$; $h = H'(\boldsymbol{pk})$;
  $J = V_A(h, y)$; $z = \hat{z} \otimes y$;
  return $\sigma = (z, \boldsymbol{c}, I, J)$;
Procedure Verify($param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \sigma$) :
  parse $\sigma = (z, \boldsymbol{c}, I, J)$; $h = H'(\boldsymbol{pk})$;
  $R'' = A(z) \odot \bigodot_{i=1}^{n} V'(pk_i, c_i)$;
  $L'' = V_A(h, z) \odot J \odot \bigodot_{i=1}^{n} V'(I, c_i)$; $c'' = \bigotimes_{i=1}^{n} c_i$;
  return $c'' = H(\boldsymbol{pk}||R''||L''||M)$;
Procedure Ext($Y, \hat{\sigma}, \sigma$) :
  parse $\hat{\sigma} = (\hat{z}, \boldsymbol{c}, I)$ *and* $\sigma = (z, \boldsymbol{c}, I, J)$; $y = z \oslash \hat{z}$;
  If $Y = A(y)$ then return y;
  return $\perp$;
Procedure Link($\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \sigma', \sigma''$) :
  parse $\sigma' = (z', \boldsymbol{c}', I', J')$ *and* $\sigma'' = (z'', \boldsymbol{c}'', I'', J'')$;
  If $I' = I''$ then return **Linked**;
  return **Unlinked**;

Then by the nature of Type-T identity authentication, it can be calculated:

$$V(\boldsymbol{pk}, \hat{z} \otimes y, \mathbf{c}^*) = A(\hat{z}) \odot A(y) \odot V'(\boldsymbol{pk}, \mathbf{c}^*) = V(\boldsymbol{pk}, \hat{z}, \mathbf{c}^*) \odot A(y)$$

Hence y could be extracted such that $A(y) = Y$. □

Theorem 4: Assuming that Type-T signature satisfy secure against special impersonation under key-only attack, $|\Delta_c| > (q_s + q_{ps})(q_h - 1) + {q_{ps}}^2 + {q_s}^2$ *and* L_R *is a hard relation, where* q_h, q_{ps}, *and* q_s *are the amount of queries to the H oracle, pre-signing oracle, and the signing oracle respectively, then the linkable dualring adaptor signature scheme is provably secure in the aEUF–CMA security model.*

Proof: In this proof, we set $\mathcal{A}$ for $\mathcal{PPT}$ who can win the aEUF-CMA security game with non-negligible probability. If such an adversary exists, algorithm $\mathcal{B}$ can be constructed, which can break the special impersonation under the key-only attack of Type-T signature or the hardness of relation L_R. Before the queries, the simulator $\mathcal{B}$ will receive the system parameter $param$ and the public key pk^*.

4) Setup: The algorithm $\mathcal{B}$ randomly samples an index $j^* \in [1, q_k]$. And then he executes $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in [1, q_k]$, $i \neq j^*$ and sets $pk_{j^*} = pk^*$. finally, this algorithm return $param$ and $S = \{pk_i\}_{i=1}^{q_k}$ to adversary $\mathcal{A}$.

5) Oracle Simulation: The algorithm $\mathcal{B}$ responds to the adversary's queries to random oracle as follows.

OR_H: The hash function H is simulated by algorithm $\mathcal{B}$ as a random oracle.

OR_Corrupt: If $i = j^*$, the $\mathcal{B}$ states failure and aborts. Otherwise, algorithm $\mathcal{B}$ responds with private key sk_i.

$\text{OR}_{\text{pre-signing}}$: The algorithm $\mathcal{B}$ outputs $\perp$ if $pk_i \notin \bar{\boldsymbol{pk}}$. If the key is asked, and its index is not j^*, $\mathcal{B}$ interacts honestly like the PreSign algorithm. Otherwise, this algorithm provides the pre-signature for pk_{j^*} as follows: Everything is identical to the PreSign algorithm, with the following exception: $\mathcal{B}$ picks random $\hat{z} \in \Delta_{\hat{z}}, c_i \in \Delta_c$ for all i, and calculates $R = V(\bar{\boldsymbol{pk}}, \hat{z}, \mathbf{c}) \odot Y = A(\hat{z}) \odot \bigodot_{i=1}^{n} V'(pk_i, c_i) \odot Y$ where $pk_i \in \bar{\boldsymbol{pk}}$, $L = V_A(h, \hat{z}) \odot \bigodot_{i=1}^{n} V'(I, c_i)$. Then the challenge value $c = H(M, \bar{\boldsymbol{pk}}, R, L) = \bigotimes_{i=1}^{n} c_i$ will be set by $\mathcal{B}$ in the random oracle. The algorithm $\mathcal{B}$ aborts if this input of hash function H has previously been queried. In the absence of this, this algorithm outputs pre-signature $\hat{\sigma} = (\hat{z}, \mathbf{c}, I)$.

OR_signing: If the L_R in this input of is NULL, $\mathcal{B}$ outputs $\perp$ when $pk_i \notin \bar{\boldsymbol{pk}}$. If the key is asked, and its index is not j^*, $\mathcal{B}$ outputs $\sigma \leftarrow \text{SIGN}(param, M, \bar{\boldsymbol{pk}}, sk_i)$ honestly. Otherwise, this algorithm randomly samples $z \in \Delta_z, c_i \in \Delta_c$ for all i, and calculates $R' = V(\bar{\boldsymbol{pk}}, z, \mathbf{c}) = A(z) \odot \bigodot_{i=1}^{n} V'(pk_i, c_i)$ where $pk_i \in \bar{\boldsymbol{pk}}$, $L' = V_A(h, z) \odot J \odot \bigodot_{i=1}^{n} V'(I, c_i)$. Then $\mathcal{B}$ will be set $c = H(M, \bar{\boldsymbol{pk}}, R', L') = \bigotimes_{i=1}^{n} c_i$ in the random oracle. If this input of H is queried previously, $\mathcal{B}$ aborts. Otherwise, the $\mathcal{B}$ returns $\sigma = (z, \mathbf{c}, I, J)$. If the content of L_R as input is (Y, y), the $\mathcal{B}$ first executes the aforementioned PreSign simulation procedure to obtain the $\hat{\sigma}$. The Adapt algorithm is then executed, yielding the σ. Finally, the $\mathcal{B}$ returns σ.

6) Challenge: $\mathcal{A}$ returns the target message $(M^*, pk_{i^*}, \bar{\boldsymbol{pk}}^*)$ to $\mathcal{B}$. Using the simulation approach described above, the algorithm $\mathcal{B}$ selects a hard relation pair (Y, y) from LockGen that hasn't been leveraged before and generates a pre-signature $\hat{\sigma}$. Then $\mathcal{B}$ returns $(\hat{\sigma}, Y)$ to $\mathcal{A}$. At the end, a linkable dualring adaptor signature $(\tilde{\boldsymbol{pk}}^*, \sigma^*)$ on M^* is forged by the adversary $\mathcal{A}$ for $((\tilde{\boldsymbol{pk}}^* \subseteq \boldsymbol{pk} \backslash \mathcal{F}) \wedge ((\star, M^*, \tilde{\boldsymbol{pk}}^*) \notin \Omega) \wedge \text{Verify}(\tilde{\boldsymbol{pk}}^*, \sigma^*, M^*) = 1)$. Among them, the $\mathcal{F}, \Omega$ is identical to that in the aSignForge experiment. The σ^* is denoted as $(z^*, \mathbf{c}^*, *)$. The following two possible cases for the adversary's forgery are discussed:

Case 1: All the components of σ^* are identical to $\sigma = \text{Adapt}(y, \hat{\sigma})$, and $\tilde{\boldsymbol{pk}}^* = \hat{\boldsymbol{pk}}^*$. This implies that adversary $\mathcal{A}$ receives the witness y, which breaks the hardness of relation L_R.

Case 2: Case 1 did not occur. In the PreSign phase, the algorithm $\mathcal{B}$ computes the corresponding $R^* = V(\tilde{\boldsymbol{pk}}^*, z^*, \mathbf{c}^*))$ first. Then he can control the random ora-

cle rewinding to the point when H is questioned for $(\tilde{\boldsymbol{pk}}^*, R^*, M^*, *)$ according to the forking lemma [28]. A different c' is returned by algorithm $\mathcal{B}$ instead. Another signature is returned by $\mathcal{A}$ as $(z', \mathbf{c}' = c_i', \cdots, c_n', *)$. Since both σ^* and σ' are usable formal signatures, We have:

$$R^* = A(z^*) \odot \bigodot_{i=1}^{n} V'(p\tilde{k}_i, c_i{}^*) = A(z') \odot \bigodot_{i=1}^{n} V'(p\tilde{k}_i, c_i{}')$$

Note that it is impossible to have $c_i{}^* = c_i{}'$ for all $i \in [1, n]$ (since $\bigotimes_{i=1}^{n} c_i{}^* \neq \bigotimes_{i=1}^{n} c_i{}'$). Algorithm $\mathcal{B}$ claims failure and terminates if $c_i{}^* = c_i{}'$. We have $c_i{}^* \neq c_i{}'$ with probability at least $1/n$. Observe that:

$$\begin{aligned} &A(z^*) \odot \bigodot_{i=1}^{n} V'(p\tilde{k}_i, c_i{}^*) \\ &= A(z^* \otimes (n-1)r \oslash \bigotimes_{i=1}^{n} z_i{}^*(i \neq j)) \odot V'(pk_j{}^*, c_j{}^*) \\ &= A(\tilde{\boldsymbol{z}}^*) \odot V'(pk_j{}^*, c_j{}^*) \end{aligned}$$

Similarly we have $A(z') \odot \bigodot_{i=1}^{n} V'(p\tilde{k}_i, c_i{}') = A(\tilde{\boldsymbol{z}}') \odot V'(pk_j{}^*, c_j{}')$ for some $\tilde{\boldsymbol{z}}'$, then algorithm $\mathcal{B}$ will output $(c_j{}^*, \tilde{\boldsymbol{z}}^*, c_i{}', \tilde{\boldsymbol{z}}')$.

Thus, the simulator $\mathcal{B}$ can extract the signer's private key $sk_j{}^*$ through two different signature pairs corresponding to the same commitment value R^* in Type-T. For example, the simulator $\mathcal{B}$ can output $sk_j = (z^* - z') \cdot (c_j{}' - c_j{}^*)^{-1}$ as a break of the secure against impersonation under key-only attack of Type-T signature in the Schnorr-based instantiation. Alternatively, $\mathcal{B}$ can output $sk_j = (z^* \cdot z'^{-1})^{(c_j{}^* - c_j{}')^{-1}}$ as a break of the secure against impersonation under key-only attack of Type-T signature in the GQ-based instantiation.

7) Probability Analysis: We examine the probability that the simulation described above will succeed (i.e., not fail).

The probability that a q_c query will succeed in the first attempt is $(1 - \frac{1}{q_k})$ for queries to the $\mathrm{OR}_{\mathrm{Corrupt}}$. The probability of success in the 2-th query is at least $(1 - \frac{1}{q_k - 1})$. The probability of success after q_c queries is at least $(1 - \frac{1}{q_k})(\frac{1}{q_k - 1}) \cdots (1 - \frac{1}{q_k - q_c + 1}) = \frac{q_k - q_c}{q_k} = 1 - \frac{q_c}{q_k}$.

For q_{ps} queries to the $\mathrm{OR}_{\mathrm{pre-signing}}$, the probability of success in the 1-th query is at least $(1 - \frac{q_h}{|\Delta_c|})$, where the amount of queries to the H oracle represents q_h. After q_{ps} queries to $\mathrm{OR}_{\mathrm{pre-signing}}$, the probability that it will succeed is no less than

$$\begin{aligned} &(1 - \frac{q_h}{|\Delta_c|})(1 - \frac{q_h + 1}{|\Delta_c|}) \cdots (1 - \frac{q_h + q_{ps} - 1}{|\Delta_c|}) \\ &\geq 1 - \frac{q_{ps}(q_h + q_{ps} - 1)}{|\Delta_c|}. \end{aligned}$$

From the simulation process of queries to the $\mathrm{OR}_{\mathrm{signing}}$, we know that its probability that it will succeed after q_s queries to $\mathrm{OR}_{\mathrm{signing}}$ is no less than $1 - \frac{q_s(q_h + q_s - 1)}{|\Delta_c|}$.

The probability of $\tilde{\boldsymbol{pk}}^* \not\subseteq \{\boldsymbol{pk} \backslash \mathcal{F}\}$ in the challenge phase is $(1 - \frac{1}{q_k - q_c})(1 - \frac{1}{q_k - q_c - 1}) \cdots (1 - \frac{1}{q_k - q_c - n + 1}) = \frac{q_k - q_c - n}{q_k - q_c}$. The probability that $\mathcal{B}$ will fail before rewinding is

$$\begin{aligned} \epsilon_B &= \epsilon(1 - \frac{q_c}{q_k})(1 - \frac{q_{ps}(q_h + q_{ps} - 1)}{|\Delta_c|}) \\ &\quad \cdot (1 - \frac{q_s(q_h + q_s - 1)}{|\Delta_c|})(1 - \frac{q_k - q_c - n}{q_k - q_c}) \\ &= \epsilon(\frac{n}{q_k})(1 - \frac{q_c}{q_k})(1 - \frac{q_{ps}(q_h + q_{ps} - 1)}{|\Delta_c|}) \\ &\quad \times (1 - \frac{q_s(q_h + q_s - 1)}{|\Delta_c|}) \end{aligned}$$

if the probability of forgery by $\mathcal{A}$ is ϵ.

If $|\Delta_c| > 8nq_h/\epsilon_B$ (it runs in time $\tau \cdot 8nq_h/\epsilon_B \cdot ln(8n/\epsilon_B)$ if $\mathcal{A}$ runs in time τ), the probability of an effective rewinding is no less than $\frac{\epsilon_B}{8}$ according to the generalized forking lemma [29]. As a result, the probability ϵ_B' of algorithm $\mathcal{B}$ breaking the special impersonation is:

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

where $2|\Delta_c| > (q_s + q_{ps})(q_h - 1) + {q_{ps}}^2 + {q_s}^2$ and $|\Delta_c| > 8nq_h/\epsilon_B$. If we take $|\Delta_c| > (q_s + q_{ps})(q_h - 1) + {q_{ps}}^2 + {q_s}^2$, the probability ϵ_B' can be further simplified. Meanwhile, if $|\Delta_c| > 16nq_h/\epsilon_B$,we have $\epsilon_B' \geq \frac{\epsilon n}{32q_k}$. □

Theorem 5: Assuming $|\Delta_c| > (q_s + q_{ps})(q_h - 1) + {q_{ps}}^2 + {q_s}^2$, where q_h, q_{ps} and q_s are the number of queries to H oracle, pre-signing oracle and signing oracle respectively, the linkable dualring adaptor signature satisfies anonymity.

Proof: Assuming that the $\mathcal{PPT}$ adversary $\mathcal{A}$ can win the experiment $\mathbf{aAnon}_{\mathcal{A}, \prod_{\mathbf{Re}, \Sigma}}$ with a non-negligible probability, we can construct an algorithm $\mathcal{B}$ with anonymity under the random oracle model.

8) Setup: $(pk_i, sk_i) \leftarrow \mathrm{KeyGen}(1^\lambda)$ for each index $i \in [1, q_k]$ are executed by algorithm $\mathcal{B}$, then he returns parameters $param$ and the set $\boldsymbol{pk}$ of public key to adversary $\mathcal{A}_1$.

9) Oracle Simulation: The following is the response from algorithm $\mathcal{B}$ to the oracle query.

OR_{H}: The random oracle H is simulated by algorithm $\mathcal{B}$.

$\mathrm{OR}_{\mathrm{pre-signing}}$: It is the same as the pre-signing oracle in the proof of Theorem 4.

$\mathrm{OR}_{\mathrm{signing}}$: It is the same as the signing oracle in the proof of Theorem 4.

10) Challenge: Adversary $\mathcal{A}_1$ returns algorithm $\mathcal{B}$ a statement Y from hard relation L_R, two indexes i_0, i_1, a collection of public keys $\hat{\boldsymbol{pk}}$ and a message M^*. According to the distribution of the output of $Z(\cdot)$, algorithm $\mathcal{B}$ randomly samples $c_1, \cdots, c_n \in \Delta_c$ and samples z from the domain of response $Delta_z$. Algorithm B generates $R = A(z) \odot \bigodot_{i=1}^{n} V'(pk_i, c_i)$, then he sets $H(M^*, \hat{\boldsymbol{pk}}, R, *) = \bigotimes_{i=1}^{n} c_i$ in the random oracle. Algorithm $\mathcal{B}$ announces failure and aborts if the hash value has already been set by the H oracle. In the absence of this, algorithm $\mathcal{B}$ sends Y and $\sigma = (c_1, \cdots, c_n, z, *)$ to adversary $\mathcal{A}_2$. In the end, algorithm $\mathcal{B}$ samples a bit b at random.

11) Output: In the end, adversary $\mathcal{A}_2$ returns a bit b'. Be aware that bit b isn't utilized in the generation of σ. Because of this, $\mathcal{A}_2$ only has a 50% chance of winning.

12) Probability Analysis: We examine the probability that the aforementioned simulation will succeed (i.e., not fail). The probability of success in the 1-th query is at least $(1 - \frac{q_h}{|\Delta_c|})$ where q_h queries to the H oracle, q_{ps} queries to the $\mathrm{OR}_{\mathrm{pre-Signing}}$ and q_s queries to the $\mathrm{OR}_{\mathrm{Signing}}$. After sending queries to all oracle, there is at least $(1 - \frac{q_s(q_h + q_s - 1)}{|\Delta_c|})(1 - \frac{q_{ps}(q_h + q_{ps} - 1)}{|\Delta_c|})$ probability of success. Assumed in this case is that $|\Delta_c| > (q_s + q_{ps})(q_h - 1) + {q_{ps}}^2 + {q_s}^2$. No $\mathcal{PPT}$ opponent can win with a non-negligible probability greater than $1/2$ if $\mathcal{B}$ does not abort. □

Theorem 6: Assuming the DL assumption is hard, the linkable dualring adaptor signature satisfies linkability w.r.t. insider corruption under the random oracle model.

Proof: Assuming that the $\mathcal{PPT}$ adversary $\mathcal{A}$ can win the experiment $\text{aLink}_{\mathcal{A},\prod_{R_e,\Sigma}}$ with a non-negligible probability, we can construct an algorithm $\mathcal{B}$ who can break the DL assumption. Suppose the instance of the hard problem that $\mathcal{B}$ desires to solve is (g, g^a).

13) Setup: Algorithm $\mathcal{B}$ executes $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for each index $i \in [1, n]$ where $pk_i^* = g^a$. $\mathcal{B}$ gives $param$ and $\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}$ to adversary $\mathcal{A}$.

14) Oracle Simulation: The following is the response from algorithm $\mathcal{B}$ to the oracle query.

$\text{OR}_{\text{corrupt}}$: If $\mathcal{A}$ asked pk_i^*, $\mathcal{B}$ fails.

$\text{OR}_{\text{pre-signing}}$: It is the same as the pre-signing oracle in the proof of Theorem 4.

$\text{OR}_{\text{signing}}$: It is the same as the signing oracle in the proof of Theorem 4.

15) Output: Adversary $\mathcal{A}$ generates two signature tuples $(\sigma_i, M_i^*, \hat{\boldsymbol{pk}}_i)_{i=1,2}$. If $\mathcal{A}$ never asked any tuple $(\star, M_i^*, \hat{\boldsymbol{pk}}_i)_{i=1,2}$ and If two pairs of unlinkable valid signatures are generated when the amount of corrupted nodes is not greater than 2, the adversary wins. According to the proof of unforgeability, $\mathcal{B}$ can rewind to the point when the R value has been generated and calculates a corrupted sk_b and another unknown private key sk_{1-b}^*. Finally, $\mathcal{B}$ returns the latter as a solution to the DL hard problem.

16) Probability Analysis: There is at least $\frac{1}{|\boldsymbol{pk}|-q_c}$ probability to complete the extraction of sk_{1-b}^* (from $pk_i^* = g^a$) for $\mathcal{B}$, where q_c is the number of queries to $\text{OR}_{\text{Corrupt}}$. □

Theorem 7: The linkable dualring adaptor signature is non-slanderable w.r.t. insider corruption in the random oracle model if DL assumption is hard.

Proof: Assuming that the $\mathcal{PPT}$ adversary $\mathcal{A}$ can win the experiment $\text{aNonFra}_{\mathcal{A},\prod_{R_e,\Sigma}}$ with a non-negligible probability, we can construct an algorithm $\mathcal{B}$ who can break the DL assumption. Suppose the instance of the hard problem that $\mathcal{B}$ desires to solve is (g, g^a).

17) Setup: Algorithm $\mathcal{B}$ executes $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)$ for each index $i \in [1, n]$ where $pk_i^* = g^a$. $\mathcal{B}$ gives $param$ and $\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}$ to adversary $\mathcal{A}$.

18) Oracle Simulation: The following is the response from algorithm $\mathcal{B}$ to the oracle query.

$\text{OR}_{\text{corrupt}}$: If $\mathcal{A}$ asked pk_i^*, $\mathcal{B}$ fails.

$\text{OR}_{\text{pre-signing}}$: It is the same as the pre-signing oracle in the proof of Theorem 4.

$\text{OR}_{\text{signing}}$: It is the same as the signing oracle in the proof of Theorem 4.

19) Output: Adversary $\mathcal{A}$ generates a signature tuple $(\sigma, M, \hat{\boldsymbol{pk}})$. If $\mathcal{A}$ forges a valid signature under the condition that the signature has not been asked and the users set $\hat{\boldsymbol{pk}}$ has not been corrupted for all, and the signature and another signature tuple $(\sigma', M', \hat{\boldsymbol{pk}}')$ is linked, then $\mathcal{A}$ wins. According to the proof of unforgeability, $\mathcal{B}$ can rewinds to the point when the R value has been generated and calculates sk_i^* such that $g^{sk_i^*} = g^a \in (\boldsymbol{pk} \cap \hat{\boldsymbol{pk}}' \cap (\hat{\boldsymbol{pk}} \backslash \mathcal{F}))$. Finally, $\mathcal{B}$ returns the sk_i^* as a solution to the DL hard problem.

20) Probability Analysis: There is at least $\frac{1}{|\boldsymbol{pk}|-q_c}$ probability to complete the extraction of sk_i^* (from $pk_i^* = g^a$) for $\mathcal{B}$, where q_c is the amount of queries to $\text{OR}_{\text{Corrupt}}$. □

B. LDRAS+: A Further Improved Scheme

In this subsection, we further combine the NISA algorithm to reduce the size of the signature as well as its computational complexity. The NISA algorithm can prove that the prover learn the tuple $\vec{a}$ such that $H(M, \boldsymbol{pk}, L, R) = \otimes_{i=1}^{n} c_i$ without exposing $\vec{a}$. Its specific optimization is aimed at the PreSign and PreVerify phases. The concrete algorithm structure of LDRAS+ is given in Algorithm 9.

Compared to the original basic LRAS, LDRAS+ incorporates the sum arguments of knowledge algorithm into the signature and verification stages. This algorithm transforms the accumulation operation of the challenge value into a proof of the knowledge tuple $\vec{a}$ and substitutes the challenge value in the pre-signature with the proof π, effectively reducing the signature size. Furthermore, in terms of computation, while the pre-signature process may experience a slight increase in computational complexity due to the NISA algorithm PF function, it proves advantageous as it significantly reduces the signature verification overhead.

VI. Performance Analysis

A. Implementation Details

This section performs a programming simulation on the protocol and tests the running time of each algorithm in the protocol. We utilize the *Multi-precision Integer and Rational Arithmetic C/C++ Library* (MIRACL) as a function calling tool in the simulation process. The implementation was based on a classic elliptic curve with its affine equation as $y^2 = x^3 + ax + b$, satisfying $\Delta = 4a^3 + 27b^2 \neq 0$. The group element size of the elliptic curve is set to be $|\mathbb{G}| = 256$ bits. The hardware environment set up for the simulation experiment in this work includes a Lenovo desktop host equipped with 16.0 GB of RAM and the central processing unit of Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz 2.90 GHz. The operating system version on the host is 64-bit Windows 11 Home Chinese Edition. For programming, Visual Studio 2022 is employed, using the C language in accordance with the ISO C11 standard. The simulation result data in this paper are the average of 100 running results of some round.

B. Evaluation

1) Communication Overhead: For the communication overhead of the PCN protocol, we measure it as the amount of information that the user needs to send when executing the interactive protocol. As shown in Table I, the communication overhead of the protocol in the Setup phase and the Lock phase is $96 + 224m$ byte and $865m$ byte respectively, and there is no communication load in other phases. For the communication overhead of LDRAS and LDRAS+, we measure it as the signature size. As shown in Table II, assuming that c and z use the same integer field $\mathbb{Z}_q$, the signature size of LRAS and LDRAS is the same and both are linear to the number of ring members positive correlation, while LDRAS+ can reduce the storage overhead of signatures from linear to logarithmic. In addition, the size of our signatures is only 1 or 2 cycles longer in group length compared to existing work. This increase is almost negligible when considering the security and functionality benefits offered by our signature.

2) Computation Time: We assess the computation time needed by the user to run the various steps of the algorithm for the PCN protocol. As shown in Table I and Figure 6,

Algorithm 9 Linkable DualRing Adaptor Signature With NISA

Procedure Setup(λ) :
  select $H : \{0,1\}^* \to \Delta_c$, $H' : \{0,1\}^* \to \mathbb{G}$;
  return $param \leftarrow$ TA.Setup(λ);
Procedure KeyGen($param$) :
  return $(pk, sk) \leftarrow$ TA.KeyGen($param$);
Procedure PreSign($param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, sk_j, Y$) :
  $\hat{\sigma} \leftarrow$ LDRAS.PreSign($param, M, \boldsymbol{pk}, sk_j, Y$);
  $\overrightarrow{a} = (c_1, \cdots, c_n)$ in $\hat{\sigma}$; store (e, L, R) in LDRAS;
  $P = L \odot A(\hat{z})^{-1} \odot R \odot V_A(h, \hat{z})^{-1} \odot Y^{-1}$;
  $\pi \leftarrow$ NISA.Proof($param, \boldsymbol{pk}, I, P, e, \overrightarrow{a}$);
  return $\hat{\sigma} = (\hat{z}, L, R, I, \pi)$;
Procedure PreVerify($param, M, \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma}, Y$) :
  parse $\hat{\sigma}$ as $(\hat{z}, L, R, I, \pi)$; $h' = H'(\boldsymbol{pk})$;
   $e' = H(\boldsymbol{pk}||R||L||M)$;
  $P' = L \odot A(\hat{z})^{-1} \odot R \odot V_A(h', \hat{z})^{-1} \odot Y^{-1}$;
  return NISA.Verify($param, \boldsymbol{pk}, I, P', e', \pi$) $= 1$;
Procedure Adapt($(Y, y), \boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \hat{\sigma}, M$) :
  parse $\hat{\sigma} = (\hat{z}, L, R, I, \pi)$; $h' = H'(\boldsymbol{pk})$; $J = h'^{y}$;
   $z = \hat{z} \otimes y$;
  return $\sigma = (z, L, R, I, \pi, J)$;
Procedure Verify($param, M, \boldsymbol{pk}, \sigma = (z, L, R, I, \pi, J)$) :
  $h'' = H'(\boldsymbol{pk})$; $e'' = H(\boldsymbol{pk}||R||L||M)$;
  $P'' = L \odot A(z)^{-1} \odot R \odot V_A(h'', z)^{-1} \odot J$;
  return NISA.Verify($param, \boldsymbol{pk}, I, P'', e'', \pi$) $= 1$;
Procedure Ext($Y, \hat{\sigma} = (\hat{z}, c_1, \cdots, c_n, I), \sigma = (z, c_1, \cdots, c_n, I, J)$) :
  $y = z \oslash \hat{z}$;
  If $Y = A(y)$ then return y;
  return $\perp$;
Procedure Link($\boldsymbol{pk} = \{pk_1, \cdots, pk_n\}, \sigma', \sigma''$) :
  parse $\sigma' = (z', L', R', I', \pi', J')$ and
   $\sigma'' = (z'', L'', R'', I'', \pi'', J'')$;
  If $I' = I''$ then return **Linked**;
  return **Unlinked**;

TABLE I
COMPARISON OF THE COST REQUIRED TO EXECUTE THE ALGORITHMS FOR THE DIFFERENT INSTANTIATIONS OF GENERIC CONSTRUCTION OF AEMHL. (WE DENOTE BY m THE NUMBERS OF INTERMEDIATE NODES)

Time(ms)\Comm(byte)	Schnorr-based	GQ-based
Setup	$2.36 + 1.28m \backslash 96 + 224m$	$2.16 + 1.21m \backslash 96 + 224m$
Lock	$68.19m \backslash 865m$	$71.99m \backslash 865m$
Rel	$0.001(m+1) \backslash 0$	$0.003(m+1) \backslash 0$
Vf	$22.74(m+1) \backslash 0$	$24.02(m+1) \backslash 0$
Link	$0.04(m+1) \backslash 0$	$0.05(m+1) \backslash 0$

GQ-based scriptless construction requires high average computational overhead. This is mainly because its signing keys are distributed in a multiplicative manner compared to the Schnorr method. Furthermore, observing that the sum of the running time of each phase does not exceed 1s when the path length of ten hops is used as the performance index [30]. This shows that the protocol performance of each instantiation is within the expected range. For signature schemes in Table II, we measure computation time by measuring computational complexity. In general, due to its special dual-ring structure, LDRAS can reduce exponent operations in the calculation process. LDRAS+ has the characteristics of signing intricately and verifying quickly because of the addition of the NISA algorithm.

3) Comparison of Underlying Solutions: Compared to existing work, the computational complexity of our LRAS pre-signature is slightly higher, but the complexity of its optimization scheme, LDRAS, is comparable and the complexity of signature verification of LDRAS+ is even lower than that of the original ring signature. Moreover, this slight efficiency gap brings obvious security and functionality advantages. As shown in the lower part of Table II, the ring signature of [6] provides only unconditional anonymity, a property that gives the users the possibility of cheating because it is not restricted. Linkability and non-slanderability can reasonably address the above problem by effectively inhibiting the behavior of malicious users and guaranteeing a certain degree of user anonymity, but the CLRAS of [18] only implements the authentication function of signature and fails to embody these aspects of the property. Additionally, the adaptation and extraction algorithm of adapter signatures in [15] can provide the function of locking the coins on the chain, but fails to provide user anonymity and anonymity-related properties.

4) Throughput: We assume that the time for AEMHL to process off-chain transactions is approximately equal to the time to execute the algorithm, including Setup(·), Lock(·), Rel(·), Vf(·), and Link(·), which is about 92 ms in total. Since typical network latency for 4G WAN and Internet connections is approximately 60 ms, a single channel in AEMHL takes approximately 152 ms to process a transaction. In addition, we assume that T is the number of channels opened on Monero, then AEMHL can increase the throughput on Monero from 1000 tps[1] to $6.58T$ tps. If AEMHL had the same scale as the Bitcoin Lightning Network (over 73,000 open channels[2]), it would have the potential to provide over 438,000 tps throughput.

In Figure III, we compare AEMHL with three state-of-the-art solutions that also have privacy protection as a design goal. Except for EC-VTS, which is not specified, other solutions all leverage the Lightning Network as the simulation topology. Since the simulation environments set up are quite different, we have standardized and unified the relevant data of each scheme. For example, for Twilight, we first converted the throughput of 820 tps per relay into 28 tps per channel and then converted it into a computational

[1]Data sourced from https://alephzero.org/blog/what-is-the-fastest-blockchain-and-why-analysis-of-43-blockchains/

[2]Data sourced from https://txstats.com/d/000000012/lightning-network?orgId=1&from=now-2d&to=now on 5 Nov, 2023

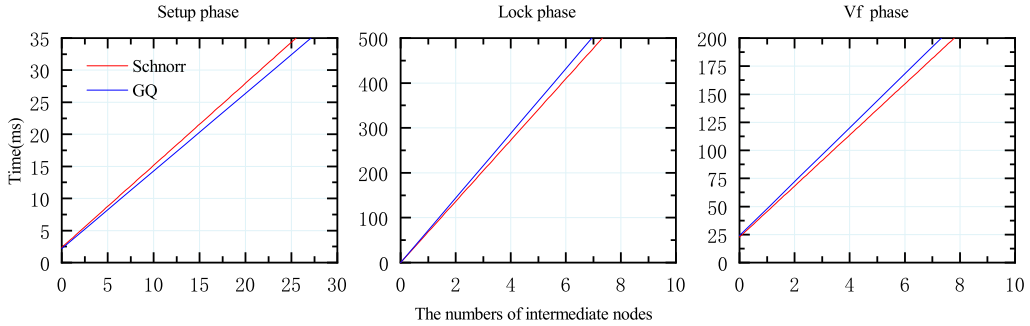

Fig. 6. Comparison of computing overhead based on Schnorr and GQ in main stages of AEMHL.

TABLE II

COMPARISON AMONG DIVERSE SIGNATURES ABOUT (PRE-)SIGNATURE COMPARISON, SECURITY AND FUNCTIONALITY. (WE ASSUME THAT $|\mathbb{G}| = 256$ BITS, $|q| = 80$ BITS, n IS THE NUMBER OF RING MEMBERS SELECTED BY BOTH PARTIES, T_M IS THE AMOUNT OF POINT MULTIPLICATION OPERATIONS, T_E IS THE AMOUNT OF POWER EXPONENTIATION OPERATIONS, A RED CROSS IS 'UNSATISFIED', A GREEN CHECKMARK IS 'SATISFIED', AND '-' IS 'CLAIMED BUT UNSATISFIED')

Schemes	(Pre-)Signature	Size	Computational complexity of (pre-)sign	Computational complexity of (pre-)verify
Schnorr-Ring [6]	$(c_1, z_1, \cdots, z_n)$	$(n+1)\|q\|$	$(n-1)T_M + (2n-1)T_E$	$nT_M + (2n)T_E$
ECDSA-Adaptor [15]	(x_K, z, K, π)	$4\|q\| + (2\log_2 n + 1)\|\mathbb{G}\|$	$2T_E$	$1T_M + 2T_E$
CLRAS [18]	$(c_1, z_1, \cdots, z_n)$	$(n+1)\|q\|$	$(n+1)T_M + 2T_E$	nT_M
Our LRAS	$(c_1, z_1, \cdots, z_n, I)$	$(n+1)\|q\| + \|\mathbb{G}\|$	$(3n-2)T_M + (4n-1)T_E$	$(3n)T_M + (4n)T_E$
Our LDRAS	$(\hat{z}, c_1, \cdots, c_n, I)$	$(n+1)\|q\| + \|\mathbb{G}\|$	$(2n-1)T_M + (2n+1)T_E$	$(2n+1)T_M + (2n+2)T_E$
Our LDRAS+	$(\hat{z}, L, R, I, \pi)$	$3\|q\| + (2\log_2 n + 3)\|\mathbb{G}\|$	$(3n + 2\log_2 n + 1)T_M + (6n + 2\log_2 n)T_E$	$(n + 2\log_2 n + 3)T_M + (n+7)T_E$

	Adaptability	Extractability	Linkability	Anonymity	Non-slanderability	Functionality
Schnorr-Ring [6]	✗	✗	✗	✓	✗	Hiding identity
ECDSA-Adaptor [15]	✓	✓	✗	✗	✗	Locking coin
CLRAS [18]	✓	✓	-	-	-	Locking coin & Hiding identity incompletely
Our LRAS	✓	✓	✓	✓	✓	Hiding identity & Locking coin
Our LDRAS	✓	✓	✓	✓	✓	Improving efficiency evenly
Our LDRAS+	✓	✓	✓	✓	✓	Providing efficient verification

TABLE III

PERFORMANCE COMPARISON OF AEMHL AND OTHER SOLUTIONS. (WE DENOTE '−' AS THE ABSENCE OF RELEVANT INFORMATION AND '⋆' AS THE VALUE ESTIMATED FOR STANDARDIZED COMPARISON)

Schemes	Topology	Setting	Computational cost(ms)	Latency(ms)	Throughput(tps)
Twilight [31]	LN	Intel SGX-1 TEE	35⋆	84	584000⋆
EC-VTS [32]	-	-	370	-	146000⋆
zk-PCN [33]	LN	Aleo 4	162	-	292000⋆
Our AEMHL	LN	4G cellular data connection	92	60	438000

cost of 35 ms. Finally, we combined the average round-trip delay of 84 ms in its simulation environment to calculate the standard transaction throughput. For EC-VTS and zk-PCN, we only unified their network delay data and estimated standard transaction throughput combined with computational overhead. Through comparison, it can be found that, except for Twilight, AEMHL's data performance in terms of computing overhead and transaction throughput is better than the other two solutions. The reason for Twilight's strength is that its TEE does not store the status of the channel, which minimizes the basis for trusted computing. In summary, compared with the latest PCNs with privacy protection, AEMHL demonstrates better system operational performance.

VII. CONCLUSION

This paper introduced a new mechanism called Anonymity-Enhancing Multi-Hop Locks (AEMHL) based on the Linkable Ring Adaptor Signature (LRAS) construction, which ensures identity anonymity for node users, maintains the fungibility of Monero, and does not rely on specific scripts underlying blockchain. In addition, we optimize LRAS to reap two improved schemes (linkable dualring adaptor signature and linkable dualring adaptor signature plus) to achieve a more efficient multi-hop lock protocol. The performance evaluation shows promising results, with Schnorr-based and GQ-based AEMHL protocols demonstrating superior performance. The optimized scheme reduces computational complexity and overhead, making the protocols suitable for different scenarios with specific requirements.

As a promising future direction, we intend to investigate the widespread implementation of our dualring-based structure to assess the advantages of efficient identity anonymization within payment channel networks at a large scale.

REFERENCES

[1] J. Poon and T. Dryja, "The Bitcoin lightning network: Scalable off-chain instant payments," Tech. Rep., 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[2] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 455–471.

[3] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_09-4_Malavolta_paper.pdf

[4] A. Poelstra, "Scriptless scripts," in *Proc. Presentation Slides*, Mar. 2017. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf

[5] P. Syverson, D Goldschlag, and M Reed, "Onion routing for anonymous and private internet connections," *Commun. ACM*, vol. 42, no. 2, p. 5, Feb. 1999.

[6] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, Nov. 2002, pp. 415–432.

[7] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 31–48.

[8] S. A. K. Thyagarajan and G. Malavolta, "Lockable signatures for blockchains: Scriptless scripts for all signatures," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 937–954.

[9] S. A. Thyagarajan, G. Malavolta, F. Schmid, and D. Schröder, "Verifiable timed linkable ring signatures for scalable payments for Monero," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2022, pp. 467–486.

[10] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding, "Dualring: Generic construction of ring signatures with efficient instantiations," in *Proc. Annu. Int. Cryptol. Conf.*, Cham, Switzerland: Springer, 2021, pp. 251–281.

[11] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "SmartPool: Practical decentralized pooled mining," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)*, 2017, pp. 1409–1426.

[12] M. Herlihy, "Atomic cross-chain swaps," in *Proc. ACM Symp. Princ. Distrib. Comput.*, Jul. 2018, pp. 245–254.

[13] H. Arslanian, "Wholesale central bank digital currencies," in *The Book of Crypto: The Complete Guide to Understanding Bitcoin, Cryptocurrencies and Digital Assets*. Berlin, Germany: Springer, 2022, pp. 185–201.

[14] L. Fournier. (2019). *One-Time Verifiably Encrypted Signatures AKA Adaptor Signatures*. [Online]. Available: URl: https://github.com/LLFourn/one-time-VES/blob/master/main. pdf.

[15] L. Aumayr et al., "Generalized channels from limited blockchain scripts and adaptor signatures," in *Proc. 27th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Singapore, Cham, Switzerland: Springer, Dec. 2021, pp. 635–664.

[16] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable data integrity," *IEEE Internet Things J.*, early access, Oct. 2023.

[17] Z. Kang, J. Li, J. Shen, J. Han, Y. Zuo, and Y. Zhang, "TFS-ABS: Traceable and forward-secure attribute-based signature scheme with constant-size," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9514–9530, Feb. 2023, doi: 10.1109/TKDE.2023.3241198.

[18] Z. Sui, J. K. Liu, J. Yu, and X. Qin, "MoNet: A fast payment channel network for scriptless cryptocurrency Monero," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2022, pp. 280–290.

[19] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology—CRYPTO'89 Proceedings 9,*. Cham, Switzerland, Springer, 1990, pp. 239–252.

[20] L. C. Guillou and J.-J. Quisquater, "A 'paradoxical' indentity-based signature scheme resulting from zero-knowledge," in *Advances in Cryptology—CRYPTO'88: Proceedings 8*. Cham, Switzerland: Springer, 1990, pp. 216–231.

[21] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre, "From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, Apr./May 2002, pp. 418–433.

[22] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, Oct. 2001, pp. 136–145.

[23] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," Tech. Rep., 2010, vol. 34. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0

[24] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009.

[25] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Prague, Czech Republic, Cham, Switzerland: Springer, May 1999, pp. 295–310.

[26] J. Katz, U. Maurer, B. Tackmann, and V. Zikas, "Universally composable synchronous computation," in *Proc. Theory Cryptogr. Conf.* Cham, Switzerland: Springer, 2013, pp. 477–498.

[27] J. Camenisch and A. Lysyanskaya, "A formal treatment of onion routing," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2005, pp. 169–187.

[28] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 390–399.

[29] A. Bagherzandi, J.-H. Cheon, and S. Jarecki, "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, Oct. 2008, pp. 449–458.

[30] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing security and privacy in decentralized credit networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss201701-5MalavoltaPaper.pdf

[31] M. Dotan, S. Tochner, A. Zohar, and Y. Gilad, "Twilight: A differentially private payment channel network," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 555–570.

[32] X. Zhou, D. He, J. Ning, M. Luo, and X. Huang, "Efficient construction of verifiable timed signatures and its application in scalable payments," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5345–5358, 2023.

[33] W. Yu, M. Xu, D. Yu, X. Cheng, Q. Hu, and Z. Xiong, "Zk-PCN: A privacy-preserving payment channel network using zk-SNARKs," in *Proc. IEEE Int. Perform., Comput., Commun. Conf. (IPCCC)*, Nov. 2022, pp. 57–64.

Xiaohu Wang is currently pursuing the M.S. degree in cyber security with the College of Computer and Cyber Security, Fujian Normal University, China. His main research interests include applied cryptography and blockchain privacy protection.

Chao Lin received the Ph.D. degree from the School of Cyber Science and Engineering, Wuhan University, in 2020. He is currently with the College of Information Science and Technology, Jinan University, China. His main research interests include applied cryptography and blockchain technology.

Xinyi Huang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently an Associate Professor with the Artificial Intelligence Thrust, Information Hub, The Hong Kong University of Science and Technology (Guangzhou). He has authored over 100 research papers in refereed international conferences and journals. His work has been cited more than 9000 times at Google Scholar (H-index: 50). His research interests include applied cryptography and network security. He is also an Associate Editor of IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has authored or coauthored more than 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and the USENIX Security Symposium. His work has been cited more than 10000 times at Google Scholar. His main research interests include cryptography and information security, in particular, and cryptographic protocols. He was a recipient of the 2018 IEEE SYSTEMS JOURNAL Best Paper Award and the 2019 *IET Information Security* Best Paper Award. He is on the editorial board of several international journals, such as *ACM Distributed Ledger Technologies: Research and Practice*, *Frontiers of Computer Science*, and IEEE TRANSACTIONS ON COMPUTERS.