A simpler construction of traceable and linkable ring signature scheme

No Author Given

No Institute Given

Abstract. Traceable and linkable ring signature scheme (TLRS) plays a major role in the construction of auditable privacy-preserving blockchains, as it empowers the auditor with traceability of signers' identities. A recent work by Li gives a modular construction of TLRS by usage of classic ring signature, one-time signature and zero-knowledge proofs, and has security against malicious auditors. In this paper, we introduce sTLRS, a simpler modification of TLRS which is constructed directly from classic ring signature, without any additional one-time signatures or zero-knowledge proofs. sTLRS has public key size reduced by 80% and verification time reduced by over 50%, compared to TLRS. Moreover, we can further modify the sTLRS to achieve anonymity, unforgeability, linkability, nonslanderability and traceability against malicious auditors.

Keywords: Auditable blockchains · Privacy preserving · Traceable and linkable ring signature · Malicious auditors.

1 Introduction

Privacy-preserving techniques in blockchain theory has been developed in this decade to provide a potential replacement of traditional blockchain-based cryptocurrencies such as Bitcoin[20] and Ethereum[6]. Privacy-preserving cryptocurrencies, represented by Monero[27] and Zerocash[24], have realized fully anonymous and confidential transactions, which can protect identities for both initiators and recipients in transactions, as well as the transaction amount, to support various privacy-preserving scenarios such as salary, donation, bidding, taxation, etc. A series of works have been proposed during these years such as Confidential Transaction[18], Mimblewimble[13], Dash[9], Monero[27] and Zerocash[24], etc. Among them, Monero uses techniques from Cryptonote[27], Ring-CT[21] and Bulletproofs[5] successively, it uses linkable ring signature scheme to hide the identity of initiator, uses Diffie-Hellman key exchange schemes[27, 21] to hide the identity of recipient and uses range proofs (Borromean[21], Bulletproofs[5]) to hide the the amount of transaction.

However, privacy-preserving cryptocurrencies are not auditable, which may cause abuse of privacy and facilitate illegal transactions by malicious users, such as money laundering, illegal asset transfer, etc. It is crucial to develop new regulatory mechanism to realize traceability of users' identities and transaction amount in privacy-preserving cryptocurrencies. To solve this issue, a recent work

by Li *et al.*[14] proposes the first fully auditable privacy-preserving blockchain against malicious auditors, their construction contains a traceable and linkable ring signature scheme (TLRS), traceable range proofs (TBoRP, TBuRP) and a traceable scheme for long-term addresses. Their work is a significant approach to overcome the regulatory barriers on privacy-preserving cryptocurrencies. As for construction of TLRS, an additional one-time signature and the validity proof of user's public key are needed to prevent slandering attack (slandering honest signers) and tracing attack (escaping from audit), both of which require extra storage and more time for generation and verification, making TLRS less efficient than the linkable ring signature (MLSAG) in Monero. So it is necessary to construct a new TLRS with simpler key generation algorithm and less computation time, to support future application of cryptocurrencies with high TPS (transactions per second).

1.1 Our Contributions

In this paper, we give a simpler construction of TLRS (named by sTLRS) by removing the one-time signature and validity proof of public key from the key generation algorithm for each user, with user's public key simplified from $PK = (g_1^x h^a, g_2^a, \pi(g_1^x h^a, g_2^a))$ to $PK = g^x$. This improvement significantly reduces the total size of PK from $(4, 1)$ to $(1, 0)$, where $(\cdot, \cdot)$ refers to number of elements in $(\mathbb{G}, \mathbb{Z}_q)$. Moreover, in sTLRS, the computation time for verification is reduced from $(7n + 1, 6n + 1)$ to $(3n + 3, 2n + 1)$, where $(\cdot, \cdot)$ refers to number of times for exponentiation and multiplication respectively (we take AOS ring signature as component for example), compared to TLRS.

In addition, we can further modify sTLRS to achieve security (anonymity, unforgeability, linkability, nonslanderability and traceability) against malicious auditors (named by sTLRS'), by adding another independent generator to the system, with the same public key as sTLRS. In sTLRS', the verification time is increased from $(3n + 3, 2n + 1)$ to $(3n + 5, 2n + 3)$, compared to sTLRS. sTLRS' is a more compact construction than the original TLRS, while maintaining the same security level (against malicious auditors).

The construction of sTLRS (sTLRS') is also modular, we can choose any suited elliptic-curve-based ring signature scheme as the component, which means that we can choose the most suited elliptic-curve-based ring signature to get the fastest (or shortest) traceable and linkable ring signature directly.

Simpler Traceable and Linkable Ring Signatures According to the design direction of [14], under similar regulatory model, we give a simpler and modular construction of traceable and linkable ring signature scheme (sTLRS) by usage of classic ring signature as component, without any additional one-time signatures or zero-knowledge proofs for public keys. Actually, in the construction of sTLRS, classic ring signature can be directly switched to traceable and linkable ring signature by randomized combination between L_{PK} and tracing keys, which will be another independent interest. We give a brief introduction of sTLRS in the following:

1. The public parameter is $(\mathbb{G}, q, g, h = g^y)$, where g (uniformly generated by system) is a generator of elliptic curve with prime order q, y is the audit trapdoor, generated by the auditor.
2. User generates his (PK, SK) by usage of public parameter, the key generation remains the same as Monero.
3. When signing, the signer publishes a tracing key TK, computes a public keys set L_{RK} for ring signature, then gets the classic ring signature τ by usage of his private key SK, the basis element (generator) for classic ring signature is different from TLRS.
4. The verifier checks whether TK is already in previous signatures to determine whether illegal signature (double spending) occurs. Then computes the classic ring signature public keys set L_{RK} and checks the validity of the classic ring signature τ, then outputs the verification results.
5. The auditor can trace the identity of signer by usage of the trapdoor y and TK.

The security of sTLRS is based on the hardness of discrete logarithm and the security of ring signature component in the random oracle model. In the following we give a brief comparison between TLRS and sTLRS:

1. sTLRS is more efficient than TLRS, no additional one-time signatures or zero-knowledge proofs are needed, the public key size is reduced by about 80%, the generation time is reduced by over 50%, and the verification time is reduced by over 50%.
2. sTLRS can also be modified to be secure against malicious auditors, which means any malicious auditor cannot double spend, slander honest users or escape from audit, same as TLRS.
3. sTLRS is adapted to Monero system, as they share the same public key generation algorithm for each UTXO. Meanwhile, the public keys do not need to regenerate when auditors change.

A concurrent work[15] gives another construction method of traceable and linkable ring signature, to achieve a traceable Monero system by making use of paring-based accumulators and signature of knowledge. Compared to [15], sTLRS has three main advantages:

1. The construction of sTLRS is modular, we can use any elliptic-based classic ring signature as component (such as AOS, Ring-CT 3.0, etc.) to achieve the most efficient (or most compact) traceable and linkable ring signature, according to different applications and parameters.
2. Security of sTLRS relies on standard assumptions (paring-free), without any trusted public parameters.
3. The tracing algorithm (as well as the generation and verification algorithms) of sTLRS is more efficient than [15].

Traceable and Linkable Multi-ring Ring Signature For Cryptocurrency Note that the key-image with fixed base h cannot be directly used in the Monero-type cryptocurrency due to the generation algorithm of UTXO's one-time public key $PK_U = g^{H(PK_v^r)}PK_s$ [27], where (PK_v, PK_s) is the public key of user, and PK_U is the one-time public key of the new UTXO. So it is necessary to construct a traceable and linkable multi-ring signature which is adaptable in the Monero-type cryptocurrency. In this paper we give a construction of traceable and linkable multi-ring signature (TRMS), following similar method of sTRLS. We will give the description of TRMS in section 6.

1.2 Related Works

Ring Signatures Ring signature is a special type of signature scheme, in which signer can sign on behalf of a group chosen by himself, while retaining anonymous within the group. In ring signature, signer selects a list of public keys L_{PK}=$\{PK_1, \cdots, PK_n\}$ as the ring elements, and uses his secret key SK_κ to sign, where $\kappa \in \{1, \cdots, n\}$. Verifier cannot determine the signer's identity. Ring signature was first proposed by Rivest, Shamir and Tauman[23] in 2001, they constructed ring signature schemes based on RSA trapdoor permutation and Robin trapdoor function, in the random oracle model. In 2002, Abe *et al.*[1] proposed AOS ring signature, which simultaneously supported discrete logarithm (via Sigma protocol) and RSA trapdoor functions (via hash and sign), also in the random oracle model. In 2006, Bender *et al.*[4] introduced the first ring signature scheme in the standard model, by making use of pairing technique. In 2015, Maxwell *et al.*[19] gave Borromean signature scheme, which is a multi-ring signature based on AOS, reduces the signature size from $mn + m$ to $mn + 1$, where n denotes the ring size and m denotes the number of rings. It's worth emphasizing that the signature sizes in these schemes are linear to the number of ring elements.

In 2004, building from RSA accumulator, Dodis *et al.*[8] proposed a ring signature scheme with constant signature size in the random oracle model. In 2007, Chandran *et al.*[7] gave a standard model ring signature scheme with $O(\sqrt{n})$ signature size, using pairing technique and CRS. In 2015, under the discrete logarithm assumption, Groth *et al.*[12] introduced a ring signature scheme with $O(\log n)$ signature size, in the random oracle model.

Linkable Ring Signatures Linkable ring signature is a variant of ring signature, in which the identity of the signer in a ring signature remains anonymous, but two ring signatures can be linked if they are signed by the same signer. Linkable ring signatures are suitable in many different practical applications such as privacy-preserving cryptocurrency (Monero), e-Voting, cloud data storage security, etc. In Monero, linkability is used to check whether double spending happens. The first linkable ring signature scheme is proposed by Liu *et al.*[17] in 2004, under discrete logarithm assumption, in the random oracle model. Later, Tsang *et al.*[26] and Au *et al.*[2] proposed accumulator-based linkable ring signatures

with constant signature size. In 2013, Yuen *et al.*[28] gave a standard model linkable ring signature scheme with $O(\sqrt{n})$ signature size, from pairing technique. In 2014, Liu *et al.*[16] gave a linkable ring signature with unconditional anonymity, he also gave the formalized security model of linkable ring signature, which we will follow in this paper. In 2015, Back *et al.*[3] proposed an efficient linkable ring signature scheme LSAG, which shortens the signature size of [17]. In 2016, based on work of Fujisaki *et al.*[10], Noether *et al.*[21] gave a linkable multi-ring signature scheme MLSAG, which supports transactions with multiple inputs, and was used by Monero. In 2017, Sun *et al.*[25] proposed Ring-CT 2.0, which is an accumulator-based linkable ring signature with asymptotic smaller signature size than Ring-CT 1.0, but is less competitive when n is small. In 2019, Yuen *et al.*[29] proposed Ring-CT 3.0, a modified Bulletproof-based 1-out-of-n proof protocol with logarithmic size, which has functionality of (linkable) ring signature and more efficient than Ring-CT 1.0 when $n > 16$. In 2019, Goodell *et al.*[11] proposed CLSAG: a modified multi-ring LRS which improves the efficiency and compactness of MLSAG.

Traceable and Linkable Ring Signatures Traceable and linkable ring signature is another variant of linkable ring signature, the identity of the signer in a ring signature can be traced by the auditor, when a signer signs two ring signatures with one secret key (illegal ring signatures), the signatures will also be linked. In 2019, Li *et al.*[15] gives a construction of traceable Monero to achieve anonymity and traceability of identities by usage of paring-based accumulators, signature of knowledge and verifiable encryption from Ring-CT 2.0, their construction provide the functionality of traceable and linkable ring signature, but relies on extra steps of verifiable encryption and decryption. Besides, in [15], traceability of long-term address depends on zk-SNARKs with CRS, which is inefficient for computation and storage, meanwhile, their work does not provide traceability of transaction amount. In 2019, TLRS[14] is proposed by Li *et al.* in the construction of the first fully auditable privacy-preserving blockchains against malicious auditors.

In this paper, we introduce sTLRS, which is a modification of TLRS to achieve better efficiency, while under standard assumptions. We also introduce sTLRS' to achieve security against malicious auditors.

1.3 Paper Organization

In section 2 we give some preliminaries; in section 3 we give the construction of the simpler traceable and linkable signature (sTLRS); in section 4 we give the security proof of sTLRS; in section 5 we introduce the modification of sTLRS to achieve security against malicious auditors; in section 6 we introduce the construction of traceable and linkable multi-ring ring signature for the Monero-type cryptocurrency; in section 7 we give the conclusion.

2 Preliminaries

2.1 Notations

In this paper, we use multiplicative cyclic group $\mathbb{G}$ to represent elliptic group with prime order $|\mathbb{G}| = q$, g is the generator of $\mathbb{G}$, group multiplication is $g_1 \cdot g_2 = g_1 g_2$ and exponentiation is g^a. We use $H(\cdot)$ to represent hash function and $negl(\cdot)$ to represent negligible functions. For verifiers, 1 is for *accept* and 0 is for *reject*. For adversaries, PPT means probabilistic polynomial time. The DDH assumption means any PPT adversary cannot distinguish (g^a, h^a) from (g^a, h^r), where r is uniformly sampled from $\mathbb{Z}_q$. The hardness of discrete logarithm problem means that any PPT adversary cannot compute x from g^x. Oracle $\mathcal{RO}$ refers to the random oracle. The security parameter of this paper is $\lambda = \lceil \log q \rceil$, where $q = |\mathbb{G}|$.

2.2 Classic Ring Signatures

Classic ring signature scheme usually consists of four algorithms: Setup, KeyGen, Rsign, and Verify.

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$ is a probabilistic polynomial time (PPT) algorithm which, on input a security parameter λ, outputs the set of security parameters Par which includes λ.
- $(PK_i, SK_i) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$ is a PPT algorithm which, on input the security parameters Par, outputs a public/private key pair (PK_i, SK_i).
- $\sigma \leftarrow \mathsf{Rsign}(SK_\kappa, \mu, L_{PK})$ is a ring signature algorithm which, on input user's secret key SK_κ, a list of users' public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, where $PK_\kappa \in L_{PK}$, and a message μ, outputs a ring signature σ.
- $1/0 \leftarrow \mathsf{Verify}(\mu, \sigma, L_{PK})$ is a verification algorithm which, on input a message μ, a list of users' public keys L_{PK} and a ring signature σ, outputs 1 or 0.

The security definition of ring signature contains *unforgeability* and *anonymity*. Before giving their definitions, we consider the following oracles which together model the ability of the adversaries in breaking the security of the schemes, in fact, the adversaries are allowed to query the four oracles below:

- $c \leftarrow \mathcal{RO}(a)$. *Random oracle*, on input a, random oracle returns a random value.
- $PK_i \leftarrow \mathcal{JO}(\perp)$. *Joining oracle*, on request, adds a new user to the system. It returns the public key PK_i of the new user.
- $SK_i \leftarrow \mathcal{CO}(PK_i)$. *Corruption oracle*, on input a public key PK_i that is a query output of $\mathcal{JO}$, returns the corresponding private key SK_i.
- $\sigma \leftarrow \mathcal{SO}(PK_\kappa, \mu, L_{PK})$. *Signing oracle*, on input a list of users' public keys L_{PK}, the public key of the signer PK_κ, and a message μ, returns a valid ring signature σ.

Definition 1 (Unforgeability) *Unforgeability for ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs* Setup *to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ wins the game if he successfully forges a ring signature $(\sigma^*, L_{PK}^*, \mu^*)$ satisfying the following:*

1. *Verify*$(\sigma^*, L_{PK}^*, \mu^*) = 1$.
2. *Every $PK_i \in L_{PK}^*$ is returned by $\mathcal{A}$ to $\mathcal{JO}$.*
3. *No $PK_i \in L_{PK}^*$ is queried by $\mathcal{A}$ to $\mathcal{CO}$.*
4. *(μ^*, L_{PK}^*) is not queried by $\mathcal{A}$ to $\mathcal{SO}$.*

We give the advantage of $\mathcal{A}$ in forging attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{forge} = \Pr[\mathcal{A} \text{ wins}].$$

A ring signature scheme is unforgeable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{forge} = negl(\lambda)$.

Definition 2 (Anonymity) *Anonymity for ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs* Setup *to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$ and $\mathcal{CO}$. $\mathcal{A}$ gives a set of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, $\mathcal{S}$ randomly picks $\kappa \in \{1, \cdots, n\}$, computes $\sigma =$ Rsign(SK_κ, μ, L_{PK}) and sends σ to $\mathcal{A}$, where SK_κ is the corresponding private key of PK_κ, then $\mathcal{A}$ outputs a guess $\kappa^* \in \{1, \cdots, n\}$. $\mathcal{A}$ wins the game if he successfully guesses $\kappa^* = \kappa$.*

We give the advantage of $\mathcal{A}$ in anonymity attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{anon} = |\Pr[\kappa^* = \kappa] - 1/n|.$$

A ring signature scheme is anonymous if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{anon} = negl(\lambda)$.

In the construction of sTLRS, we use classic ring signature (unforgeable and anonymous in the random oracle model, simulatable by programming the random oracle) as component, we may select AOS scheme[1] (linear size) or Ring-CT 3.0[29] (logarithmic size) in our construction. The choice of classic ring signature component is not restricted, we can choose the most suited ones (most efficient or most compact ones) for different ring sizes in different applications, we omit the detailed description of these ring signatures for brevity.

2.3 Linkable Ring Signatures

Compared to classic ring signature, linkable ring signature has the function of linkability, that is, when two ring signatures are signed by the same signer, they are linked by the algorithm Link. We give the definition of Link below:

- *linked/unlinked* $\leftarrow$ Link$((\sigma, \mu, L_{PK}), (\sigma', \mu', L'_{PK}))$: verifier checks the two ring signatures are linked or not, output the result.

The security definition of linkable ring signature contains *unforgeability*, *anonymity*, *linkability* and *nonslanderability*. The *unforgeability* is the same as Definition 1, and the *anonymity* is slightly different from Definition 2 with additional requirements that all public keys in L_{PK} are returned by $\mathcal{A}$ to $\mathcal{JO}$ and all public keys in L_{PK} are not queried by $\mathcal{A}$ to $\mathcal{CO}$ (if the adversary corrupts some of the public keys, then he can break the anonymity of the scheme by compute the corresponding key-images in advance). In the rest of this paper, we use this modified definition of *anonymity* in sTLRS and its security proof.

We give the definition of *linkability* and *nonslanderability* as follows:

Definition 3 (Linkability) *Linkability for linkable ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ wins the game if he successfully forges k ring signatures $(\sigma_i, L^i_{PK}, \mu_i), i = 1, \cdots, k$, satisfying the following:*

1. *All σ_is are not returned by $\mathcal{A}$ to $\mathcal{SO}$.*
2. *All L^i_{PK} are returned by $\mathcal{A}$ to $\mathcal{JO}$.*
3. $\mathsf{Verify}(\sigma_i, L^i_{PK}, \mu_i) = 1, i = 1, \cdots, k$.
4. *$\mathcal{A}$ queried $\mathcal{CO}$ less than k times.*
5. $\mathsf{Link}((\sigma_i, L^i_{PK}, \mu_i), (\sigma_j, L^j_{PK}, \mu_j)) =$ *unlinked for* $i, j \in \{1, \cdots, k\}$ *and* $i \neq j$.

We give the advantage of $\mathcal{A}$ in link attack as follows:

$$\mathrm{Adv}^{link}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins}].$$

A linkable ring signature scheme is linkable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{link}_{\mathcal{A}} = negl(\lambda)$.

The *nonslanderability* of a linkable ring signature scheme is that $\mathcal{A}$ cannot slander other honest users by generating a signature linked with signatures from honest users. In the following we give the definition:

Definition 4 (Nonslanderability) *Nonslanderability for linkable ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ gives a list of public keys L_{PK}, a message μ and a public key $PK_\kappa \in L_{PK}$ to $\mathcal{S}$, $\mathcal{S}$ returns the corresponding signature $\sigma \leftarrow \mathsf{Rsign}(SK_\kappa, L_{PK}, \mu)$ to $\mathcal{A}$. $\mathcal{A}$ wins the game if he successfully outputs a ring signature $(\sigma^*, L^*_{PK}, \mu^*)$, satisfying the following:*

1. $\mathsf{Verify}(\sigma^*, L^*_{PK}, \mu^*) = 1$.
2. *PK_κ is not queried by $\mathcal{A}$ to $\mathcal{CO}$.*
3. *PK_κ is not queried by $\mathcal{A}$ as input to $\mathcal{SO}$.*
4. $\mathsf{Link}((\sigma, L_{PK}, \mu), (\sigma^*, L^*_{PK}, \mu^*)) =$ *linked.*

We give the advantage of $\mathcal{A}$ in slandering attack as follows:

$$\mathrm{Adv}^{slander}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins}].$$

A linkable ring signature scheme is nonslanderable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{slander}_{\mathcal{A}} = negl(\lambda)$.

According to [16], linkability and nonslanderability imply unforgeability:

Lemma 5 ([16]) *If a linkable ring signature scheme is linkable and nonslanderable, then it is unforgeable.*

2.4 Traceable and Linkable Ring Signatures

Similar to the security definitions of linkable ring signature, a PPT adversary $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$, the security of TLRS contains unforgeability, anonymity, linkability, nonslanderability and traceability. Considering the existence of auditor, who can trace the identities of signers, so the anonymity only holds for someone not possesses the trapdoor. Moreover, the unforgeability, linkability, nonslanderability remain the same as in linkable ring signature, even for malicious auditor (or adversary who corrupts the auditor), he cannot forge signatures of other users or break the linkability and nonslanderability of TLRS, which means that malicious auditor cannot spend money of other users, double spend or slander other honest users.

Traceability enables auditor with ability to trace signers' identities, for any PPT adversary $\mathcal{A}$ with possession of trapdoor, he cannot escape from audit. We give the formal definition of traceability as follows:

Definition 6 (Traceability) *Traceability for traceable and linkable ring signature schemes (TLRS) is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide the public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$. $\mathcal{A}$ generates a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, $\mathcal{A}$ wins the game if he successfully generates a valid TLRS signature (σ, L_{PK}, μ) using $PK_\kappa \in L_{PK}$, satisfying the following:*

1. $\mathsf{Verify}(\sigma, L_{PK}, \mu) = 1$.
2. $PK_i \neq PK_j$ *for* $1 \leq i < j \leq n$.
3. $\mathsf{Trace}(\sigma, y) \neq \kappa$ *or* $\mathsf{Trace}(\sigma, y) = \perp$.

We give the advantage of $\mathcal{A}$ in tracing attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{trace} = \Pr[\mathcal{A} \text{ wins}].$$

TLRS scheme is traceable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{trace} = negl(\lambda)$.

We introduce the construction of TLRS[14] for single ring in the following:

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$:
  1. System chooses an elliptic curve $\mathbb{G}$ and generators $g_1, g_2 \in \mathbb{G}$ independently, the auditor generates $y \in \mathbb{Z}_q$ as the trapdoor, computes $h = g_2^y$, system outputs $(\mathbb{G}, q, g_1, g_2, h)$ as the public parameters, in which the auditor dose not know the relation between g_1 and h.
- $(PK, SK) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$:
  1. According to the public parameters $(\mathbb{G}, q, g_1, g_2, h)$, user Alice samples $x, a \in \mathbb{Z}_q$, computes $\mathsf{rpk} = g_1^x h^a, TK = g_2^a, \mathsf{opk} = h^a$;

2. Alice gives the validity proof $\pi(\mathsf{rpk}, TK) = \pi_{Swit}(g_1^x h^a, g_1^x (g_2 h)^a)$, that is, she gives the switch proof between $\mathsf{rpk} = g_1^x h^a$ and $\mathsf{rpk} \cdot TK = g_1^x (g_2 h)^a$ that they share the same exponents $(x = x, a = a)$ with basis (g_1, h) and $(g_1, g_2 h)$;
3. Alice outputs $PK = (\mathsf{rpk}, TK, \pi(\mathsf{rpk}, TK))$, and retains $SK = (\mathsf{rsk} = x, \mathsf{osk} = a)$.

– $\sigma \leftarrow \mathsf{Sign}(SK_\kappa, \mu, L_{PK})$:
1. For a message μ, Alice chooses another $n-1$ users, together with her own public key, to generate a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, where Alice's $PK = PK_\kappa \in L_{PK}, \kappa \in \{1, \cdots, n\}$;
2. Alice outputs $\mathsf{opk} = h^{a_\kappa}$, then computes

$$L_{\mathsf{rpk}} = \{\mathsf{rpk}_1 \cdot \mathsf{opk}^{-1}, \cdots, \mathsf{rpk}_n \cdot \mathsf{opk}^{-1}\}$$
$$= \{g_1^{x_1} h^{a_1 - a_\kappa}, \cdots, g_1^{x_n} h^{a_n - a_\kappa}\};$$

3. Alice runs ring signature $\tau_1 \leftarrow \mathsf{Rsign}(\mathsf{rsk}, \mu, L_{\mathsf{rpk}}, \mathsf{opk})$ using L_{rpk} and $\mathsf{rsk} = x_\kappa$, outputs τ_1;
4. Alice runs one-time signature $\tau_2 \leftarrow \mathsf{Osign}(\mathsf{osk}, \tau_1, \mathsf{opk})$ using $\mathsf{opk} = h^{a_\kappa}$ and $\mathsf{osk} = a_\kappa$ (h is the generator);
5. Alice outputs $\sigma = (\tau_1, \tau_2, \mu, L_{PK}, \mathsf{opk})$.

– $1/0 \leftarrow \mathsf{Verify}(\tau_1, \tau_2, \mu, L_{PK}, \mathsf{opk})$:
1. Verifier checks the validity of $\pi(\mathsf{rpk}_i, TK_i)$ for every $1, \cdots, n$;
2. Verifier computes $L_{\mathsf{rpk}} = \{\mathsf{rpk}_1 \cdot \mathsf{opk}^{-1}, \cdots, \mathsf{rpk}_n \cdot \mathsf{opk}^{-1}\}$;
3. Verifier checks the validity of the ring signature τ_1 and the one-time signature τ_2;
4. If all passed then outputs 1, otherwise outputs 0.

– *linked/unlinked* $\leftarrow \mathsf{Link}(\sigma, \sigma')$:
1. For two TLRS signatures $\sigma = (\tau_1, \tau_2, \mu, L_{PK}, \mathsf{opk})$ and $\sigma' = (\tau_1', \tau_2', \mu', L_{PK}', \mathsf{opk}')$, if $\mathsf{opk} = \mathsf{opk}'$ then verifier outputs *linked*, otherwise outputs *unlinked*.

– $\kappa^* / \perp \leftarrow \mathsf{Trace}(\sigma, y)$:
1. For $\sigma = (\tau_1, \tau_2, \mu, L_{PK}, \mathsf{opk})$, the auditor extracts $TK_1, \cdots, TK_n$ from L_{PK}, computes TK_i^y for $i = 1, \cdots, n$, outputs the smallest κ^* such that $\mathsf{opk} = TK_{\kappa^*}^y$ as the trace result, otherwise outputs $\perp$.

TLRS achieves anonymity, unforgeability, linkability, nonslanderability and traceability against malicious auditors.

3 Simpler Traceable and Linkable Ring Signature

In this section, we give the construction of simpler traceable and linkable ring signature scheme (sTLRS) by modifying the key generation algorithm KeyGen, signature algorithm Sign and removing the additional one-time signature and zero-knowledge proofs to achieve better efficiency compared to TLRS. sTLRS achieves unforgeability, anonymity, linkability, nonslanderability and traceability. In the scenario of privacy-preserving cryptocurrencies, unforgeability works

for security of users' accounts, anonymity works for anonymity of signers' identities, linkability and nonslanderability works for prevention of double-spending (actively or passively), traceability works for unconditional audit of signers' identities.

3.1 Construction

In our construction of sTLRS, we use classic ring signature (AOS, AOS' or RingCT 3.0) as the ring signature component. Actually, we assume these schemes are anonymous and unforgeable in the random oracle model, which makes sTLRS secure under standard assumptions. We give the introduction of sTLRS in the following (single ring as example):

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$:
  1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, the auditor generates $y \in \mathbb{Z}_q$ as the trapdoor, computes $h = g^y$, system outputs $(\mathbb{G}, q, g, h)$ as the public parameters.
- $(PK, SK) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$:
  1. According to the public parameters $(\mathbb{G}, q, g, h)$, user Alice samples $x \in \mathbb{Z}_q^*$ as her secret key, then computes $PK = g^x$;
  2. Alice outputs $PK = g^x$, and retains $SK = x$.
- $\sigma \leftarrow \mathsf{Sign}(SK_\kappa, \mu, L_{PK})$:
  1. For a message μ, Alice chooses another $n-1$ users, together with her own public key, to generate a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, where Alice's $PK = PK_\kappa \in L_{PK}, \kappa \in \{1, \cdots, n\}$;
  2. Alice outputs $TK = h^{x_\kappa}$, then computes $e_1 = H(L_{PK}, TK, 1)$ and $e_2 = H(L_{PK}, TK, 2)$;
  3. Alice computes and outputs
  $$L_{RK} = \{PK_1^{e_1} \cdot TK^{e_2}, \cdots, PK_n^{e_1} \cdot TK^{e_2}\}$$
  $$= \{g^{e_1 x_1} h^{e_2 x_\kappa}, \cdots, g^{e_1 x_n} h^{e_2 x_\kappa}\};$$
  4. Alice runs classic ring signature $\tau \leftarrow \mathsf{Rsign}(SK, \mu, L_{RK}, TK)$ using L_{RK} and $SK = x_\kappa$, outputs τ ($g^{e_1} h^{e_2}$ as the generator);
  5. Alice outputs $\sigma = (\tau, \mu, L_{PK}, TK)$.
- $1/0 \leftarrow \mathsf{Verify}(\tau, \mu, L_{PK}, TK)$:
  1. Verifier computes $e_1^* = H(L_{PK}, TK, 1)$ and $e_2^* = H(L_{PK}, TK, 2)$;
  2. Verifier computes $L_{RK}^* = \{PK_1^{e_1^*} \cdot TK^{e_2^*}, \cdots, PK_n^{e_1^*} \cdot TK^{e_2^*}\}$;
  3. Verifier checks the validity of ring signature τ ($g^{e_1^*} h^{e_2^*}$ as the generator);
  4. If all passed then outputs 1, otherwise outputs 0.
- *linked/unlinked* $\leftarrow \mathsf{Link}(\sigma, \sigma')$:
  1. For two valid sTLRS signatures $\sigma = (\tau, \mu, L_{PK}, TK)$ and $\sigma' = (\tau', \mu', L'_{PK}, TK')$, if $TK = TK'$ then verifier outputs *linked*, otherwise outputs *unlinked*.
- $\kappa^* / \perp \leftarrow \mathsf{Trace}(\sigma, y)$:
  1. For $\sigma = (\tau, \mu, L_{PK}, TK)$, the auditor extracts $PK_1, \cdots, PK_n$ from L_{PK}, computes PK_i^y for $i = 1, \cdots, n$, outputs the smallest $\kappa^* \in \{1, \cdots, n\}$ such that $TK = PK_{\kappa^*}^y$ as the trace result, otherwise outputs $\perp$.

3.2 Correctness

Theorem 7 (Correctness of sTLRS) *For an honest user Alice in sTLRS, she can complete the traceable and linkable ring signature successfully, and the behavior of double signing (double spending) will be detected while the identity of Alice remaining anonymous. Moreover, the auditor can trace her identity correctly.*

Proof. In sTLRS, for Alice's public key $PK = PK_\kappa = g^{x_\kappa}$, she can compute $TK = h^{x_\kappa}$ and e_1, e_2, then she can compute $L_{RK} = \{g^{e_1 x_1} h^{e_2 x_\kappa}, \cdots, g^{e_1 x_n} h^{e_2 x_\kappa}\}$. Since $g^{e_1 x_\kappa} h^{e_2 x_\kappa} = (g^{e_1} h^{e_2})^{x_\kappa}$, then Alice can use her secret key $SK = x_\kappa$ to generate the classic ring signature τ using $g^{e_1} h^{e_2}$ as the generator.

When double signing occurs, we know from the linkability of sTLRS that Alice must have used $TK = h^{x_\kappa}$ for twice (proved in *Theorem* 9), then the verifier can detect that double signing occurs and outputs *linked*, at the same time, anyone (except for the auditor) cannot learn any information about the identity of signer by the anonymity of sTLRS (proved in *Theorem* 8).

For the auditor, he can compute $PK_\kappa^y = g^{y x_\kappa} = h^{x_\kappa} = TK$ and then outputs $\mathsf{Trace}(\sigma, y) = \kappa$ correctly. □

3.3 Applications in Blockchain

In the applications of privacy-preserving blockchains, under UTXO model, the $PK = g^x$ can be regarded as the UTXO public key generated in the last transaction, which will be published as the UTXO public key $PK = g^x$ during the last transaction. When making transactions, the UTXO owner runs the sTLRS scheme to hide the identity of the real UTXO, he also outputs $TK = h^x$, which is regarded as the Key-image of the UTXO, and Link is used for detection of double spending. Trace is used for tracing signers' identities by the auditor, which brings the regulatory function to the blockchains.

4 Security proofs

In this section we give the security proofs of sTLRS, including anonymity, unforgeability, linkability, nonslanderability and traceability. The security of sTLRS only holds for adversary who does not possess the trapdoor.

4.1 Proof of Anonymity

Theorem 8 (Anonymity) *sTLRS is anonymous for any PPT adversary $\mathcal{A}$ (without possession of trapdoor), assuming the classic ring signature is simulatable by programming the random oracle in the random oracle model.*

Proof. Assume $\mathcal{A}$ is playing the game with $\mathcal{S}$ in Definition 2, $\mathcal{A}$ he generates a message μ and a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, where $PK_i = g^{x_i}$, and all PK_is are returned by $\mathcal{JO}$, and $\mathcal{S}$ knows all $SK_i = x_i$.

We consider the following games between $\mathcal{S}$ and $\mathcal{A}$:

- **Game 0**. $\mathcal{S}$ samples $\kappa \in \{1, \cdots, n\}$ uniformly at random, publishes $TK = h^{x_\kappa}$, computes $e_1 = H(L_{PK}, TK, 1)$, $e_2 = H(L_{PK}, TK, 2)$ and $L_{RK} = \{g^{e_1 x_1} h^{e_2 x_\kappa}, \cdots, g^{e_1 x_n} h^{e_2 x_\kappa}\}$, generates the classic ring signature $\tau = \mathsf{Rsign}(SK, \mu, L_{RPK}, TK)$, outputs $\sigma = (\tau, \mu, L_{PK}, TK)$ to $\mathcal{A}$. When $\mathcal{A}$ receives σ, he gives a guess $\kappa^* \in \{1, \cdots, n\}$.
- **Game 1**. $\mathcal{S}$ samples $\kappa \in \{1, \cdots, n\}, r \in \mathbb{Z}_q$ uniformly at random,, publishes $TK = h^r$, computes $e_1 = H(L_{PK}, TK, 1)$, $e_2 = H(L_{PK}, TK, 2)$ and $L_{RK} = \{g^{e_1 x_1} h^{e_2 r}, \cdots, g^{e_1 x_n} h^{e_2 r}\}$, generates the classic ring signature $\tau = \mathsf{Rsign}(\mu, L_{RK}, TK)$ by programming the random oracle, outputs $\sigma = (\tau, \mu, L_{PK}, TK)$ to $\mathcal{A}$. When $\mathcal{A}$ receives σ, he gives a guess $\kappa^* \in \{1, \cdots, n\}$.

In the two games above, Game 0 is the real game between $\mathcal{S}$ and $\mathcal{A}$ in sTLRS, and Game 1 is the simulated game in the random oracle model. In game 1, r is uniformly sampled by $\mathcal{S}$, which is statistical independent from the L_{PK}, then $\mathsf{Pr}_{\mathcal{A}}[\kappa^* = \kappa] = 1/n$.

Then we only need to prove that game 0 and game 1 are computational indistinguishable. If fact, the differences between the two games are the generations of TK and L_{RK}. According to DDH assumption, $(g, h, g^{x_\kappa}, h^{x_\kappa})$ and $(g, h, g^{x_\kappa}, h^r)$ are computational indistinguishable, then $\mathcal{A}$ cannot distinguish h^{x_κ} (in game 0) from h^r (in game 1). Then we know $\mathcal{A}$ cannot distinguish $\{g^{e_1 x_1} h^{e_2 x_\kappa}, \cdots, g^{e_1 x_n} h^{e_2 x_\kappa}\}$ from $\{g^{e_1 x_1} h^{e_2 r}, \cdots, g^{e_1 x_n} h^{e_2 r}\}$, then we know game 0 and game 1 are computational indistinguishable, then we finish the anonymity proof of sTLRS. □

4.2 Proof of Linkability

Theorem 9 (Linkability) *sTLRS is linkable for any PPT adversary $\mathcal{A}$ (without possession of trapdoor), assuming the unforgeability of ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ without possession of the trapdoor y, when $\mathcal{A}$ finished the link game with $\mathcal{S}$ in Definition 3, we assume that $\mathcal{A}$ wins the link game with nonnegligible advantage δ, that is, $\mathcal{A}$ returned k sTLRS signatures $\sigma_i = (\tau_i, \mu_i, L^i_{PK}, TK_i), i = 1, \cdots, k$ (τ_is are the classic ring signatures), satisfying the following requirements:

1. All $\sigma_i, i = 1, \cdots, k$ are not returned by $\mathcal{SO}$.
2. All public keys from $L^i_{PK}, i = 1, \cdots, k$ are returned by $\mathcal{JO}$.
3. $\mathsf{Verify}(\sigma_i, L^i_{PK}, \mu_i) = 1$ for $i = 1, \cdots, k$.
4. $\mathcal{A}$ queried $\mathcal{CO}$ less than k times.
5. $\mathsf{Link}((\sigma_i, L^i_{PK}, \mu_i), (\sigma_j, L^j_{PK}, \mu_j)) = \textit{unlinked}$ for $i \neq j \in \{1, \cdots, k\}$.

We first prove a statement that, for a list of users' public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$ returned by $\mathcal{JO}$ with $PK_i = g^{x_i}$, any PPT adversary $\mathcal{A}$ generates a valid sTLRS signature $\sigma \nleftarrow \mathcal{SO}$ if and only if he quires the $\mathcal{CO}$ at least once, except for negligible probability $\epsilon_0 = negl(\lambda)$.

- $\Rightarrow$. If $\mathcal{A}$ gets $SK = x_i$ from $\mathcal{CO}$, and then $\mathcal{A}$ can run the sTLRS signature scheme to generate a valid signature $\sigma = (\tau, \mu, L_{PK}, TK)$.
- $\Leftarrow$. Assume $\mathcal{A}$ did not query the $\mathcal{CO}$ and $\mathcal{SO}$ for $L_{PK} = \{PK_1, \cdots, PK_n\}$ and finished the sTLRS signature over $L_{PK} = \{PK_1, \cdots, PK_n\}$ with non-negligible probability δ_1. We first prove that $\mathcal{A}$ does not know any of the secret keys in L_{PK}. Actually, under the hardness of discrete logarithm, $\mathcal{A}$ cannot compute x_i from $PK_i = g^{x_i}, i = 1, \cdots, n$, then the probability of $\mathcal{A}$ obtaining any of x_i is $\epsilon_1 = negl(\lambda)$.

  Next, according to the assumption that $\mathcal{A}$ generates a valid signature $\sigma = (\tau, \mu, L_{PK}, TK)$, then he must have finished the classic signature τ (with generator $g^{e_1}h^{e_2}$), where $e_1 = H(L_{PK}, TK, 1)$, $e_2 = H(L_{PK}, TK, 2)$. Without loss of generality, we assume $TK = g^s h^t$ output by $\mathcal{A}$, then we have $L_{RK} = \{g^{e_1 x_1}(g^s h^t)^{e_2}, \cdots, g^{e_1 x_n}(g^s h^t)^{e_2}\}$. Since the classic ring signature scheme achieves unforgeability, and $\mathcal{A}$ finished the classic ring signature τ with L_{RK} under generator $g^{e_1}h^{e_2}$, then we get $\mathcal{A}$ knows $SK = z$ for at least one $i \in \{1, \cdots, n\}$ *s.t.* $g^{e_1 x_i}(g^s h^t)^{e_2} = (g^{e_1}h^{e_2})^z$, except for negligible probability $\epsilon_2 = negl(\lambda)$. We can also assume that $e_1 = 0$ or $e_2 = 0$ happens with negligible probability $\epsilon_3 = negl(\lambda)$, which means $\mathcal{A}$ gets a solution for $g^{e_1(x_i - z) + e_2 s} = h^{e_2(z-t)}$ with nonnegligible probability $\delta_1 - \epsilon_1 - \epsilon_2 - \epsilon_3$, if $t \neq z$, then this contradicts with the hardness of discrete logarithm problems, so we have $t = z$. Then we have $(x_i - t)e_1 + se_2 = 0$, if $s \neq 0$, then $e_2 = e_1 s^{-1}(t - x_i)$, which means e_2 is pre-computed before $\mathcal{A}$ runs the hash function (random oracle), which happens with negligible probability. Then we get $s = 0, z = t = x_i$, which contradicts to the assumptions above. Then we get that $\mathcal{A}$ generates a valid sTLRS signature $\sigma \nleftarrow \mathcal{SO}$ if and only if he quires the $\mathcal{CO}$ at least once, except for negligible probability.

According to the fourth requirement that the number of times of $\mathcal{A}$ querying $\mathcal{CO}$ is $\leq k - 1$, and $\mathcal{A}$ returned k valid sTLRS signatures $\sigma_i = (\tau_i, \mu_i, L^i_{PK}, TK_i)$, $i = 1, \cdots, k$, then we know there are two sTLRS signatures from the same query of $\mathcal{CO}$, saying $SK = z$ from $PK = g^z$, and $\mathcal{A}$ finished two unlinked valid sTLRS signature, then there is at least one $TK_i = g^s h^t \neq h^z$ from the two sTLRS signatures (otherwise they will be linked). We have $L_{RK} = \{g^{e_1 x_1}(g^s h^t)^{e_2}, \cdots, g^{e_1 x_n}(g^s h^t)^{e_2}\}$, since $\exists j \in \{1, \cdots, n\}$ *s.t.* $x_j = z$, and $\mathcal{A}$ signs with PK_j, then we have $g^{e_1 x_j}(g^s h^t)^{e_2} = (g^{e_1}h^{e_2})^t g^{e_1(z-t)+e_2 s}$ with $g^s h^t \neq h^z$, if $e_1(z - t) + e_2 s = 0$, then we have $z = t$ and $s = 0$, otherwise e_1 (or e_2) is pre-computed before $\mathcal{A}$ runs the hash function (random oracle), which happens with negligible probability ϵ_1. Then we get $e_1(z - t) + e_2 s \neq 0$, and this means $\mathcal{A}$ can compute x *s.t.* $(g^{e_1}h^{e_2})^x = (g^{e_1}h^{e_2})^t g^{e_1(z-t)+e_2 s}$, otherwise $\mathcal{A}$ will break the unforgeability of classic ring signature, which happens with negligible probability ϵ_2, however, we know that $(g^{e_1}h^{e_2})^x = (g^{e_1}h^{e_2})^t g^{e_1(z-t)+e_2 s}$ implies a non-trivial relationship between g and h, which happens with nonnegligible probability $\delta - k\epsilon_0 - \epsilon_1 - \epsilon_2$, this contradicts to the hardness assumption of discrete logarithm problem, then we finish the linkability proof of sTLRS. □

4.3 Proof of Nonslanderability

Theorem 10 (Nonslanderability) *sTLRS is nonslanderable for any PPT adversary $\mathcal{A}$ (without possession of trapdoor), assuming the unforgeability of classic ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ without possession of the trapdoor y, when $\mathcal{A}$ finished the slandering game with $\mathcal{S}$ in Definition 4, $\mathcal{A}$ gave a list of public keys L_{PK}, a message μ and a public key $PK_\kappa \in L_{PK}$ to $\mathcal{S}$, $\mathcal{S}$ returns the corresponding signature $\sigma \leftarrow \mathsf{Sign}(SK_\kappa, L_{PK}, \mu)$ to $\mathcal{A}$. We assume that $\mathcal{A}$ wins the slandering game with nonnegligible advantage δ, that is, $\mathcal{A}$ successfully outputs a ring signature $\sigma^* = (\tau^*, \mu^*, L^*_{PK}, TK^*)$, satisfying the following:

1. $\mathsf{Verify}(\sigma^*, L^*_{PK}, \mu^*) = 1$.
2. PK_κ is not queried by $\mathcal{A}$ to $\mathcal{CO}$.
3. PK_κ is not queried by $\mathcal{A}$ as input to $\mathcal{SO}$.
4. $\mathsf{Link}((\sigma, L_{PK}, \mu), (\sigma^*, L^*_{PK}, \mu^*)) = linked$.

From the definition of Link, we know that $TK^* = TK = h^{x_\kappa}$, since $PK_\kappa = g^{x_\kappa}$ was not queried by $\mathcal{A}$ to $\mathcal{CO}$ and $\mathcal{SO}$, then $\mathcal{A}$ does not know $SK = x_\kappa$ except for negligible probability $\epsilon_0 = negl(\lambda)$ under the hardness of discrete logarithm problems. Then we know $\mathcal{A}$ successfully produced a classic ring signature τ^* with nonnegligible advantage $\delta - \epsilon_0$, according to the unforgeability of classic ring signature, we know that $\mathcal{A}$ knows at least one signing key except for negligible probability ϵ_1, that is, there exists $j \in \{1, \cdots, n\}$, $\mathcal{A}$ knows x *s.t.* $(PK^*_j)^{e_1} TK^{e_2} = (g^{e_1} h^{e_2})^x$ with nonnegligible advantage $\delta - \epsilon_0 - \epsilon_1$, where $e_1 = H(L^*_{PK}, TK, 1)$, $e_2 = H(L^*_{PK}, TK, 2)$. Without loss of generality, we assume $PK^*_j = g^s h^t$ output by $\mathcal{A}$, then we have $(g^s h^t)^{e_1} h^{e_2 x_\kappa} = (g^{e_1} h^{e_2})^x = (g^{e_1} h^{e_2})^s h^{e_1 t + e_2(x_\kappa - s)}$, using similar arguments in Theorem 9, if $e_1 t + e_2(x_\kappa - s) = 0$, then we have $x_\kappa = s$ and $t = 0$, otherwise e_1 (or e_2) is pre-computed before $\mathcal{A}$ runs the hash function (random oracle), which happens with negligible probability ϵ_2. Then $e_1 t + e_2(x_\kappa - s) \neq 0$ and $\mathcal{A}$ gets a non-trivial relationship between g and h with nonnegligible advantage $\delta - \epsilon_0 - \epsilon_1 - \epsilon_2$, which contradicts to the hardness of discrete logarithm problems, then we finish the nonslanderability proof of sTLRS. □

According to lemma 5, we get the unforgeability of sTLRS:

Corollary 11 (Unforgeability) *sTLRS is unforgeable for any PPT adversary $\mathcal{A}$ without possession of trapdoor.*

4.4 Proof of Traceability

Theorem 12 (Traceability) *sTLRS is traceable for any PPT adversary $\mathcal{A}$ (without possession of trapdoor), assuming the unforgeability of classic ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ without possession of the trapdoor y, when $\mathcal{A}$ finished the tracing game with $\mathcal{S}$ in Definition 6, $\mathcal{A}$ generates a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, we assume that $\mathcal{A}$ wins the tracing game with nonnegligible advantage δ, that is, $\mathcal{A}$ generates a sTLRS signature $\sigma = (\tau, \mu, L_{PK}, TK)$ using $PK_\kappa \in L_{PK}$, satisfying the following:

1. $\mathsf{Verify}(\sigma, L_{PK}, \mu) = 1$.
2. $PK_i \neq PK_j$ for $1 \leq i < j \leq n$.
3. $\mathsf{Trace}(\sigma, y) \neq \kappa$ or $\mathsf{Trace}(\sigma, y) = \perp$.

It should be emphasized that the OPK in TLRS is actually TK in sTLRS. We assume $PK_i = g^{x_i}h^{y_i}, i = 1, \cdots, n$ returned by $\mathcal{A}$ without loss of generality, and assume $TK = g^s h^t$. Then we have:

$$\begin{aligned} L_{RK} &= \{(g^{x_1}h^{y_1})^{e_1}(g^s h^t)^{e_2}, \cdots, (g^{x_n}h^{y_n})^{e_1}(g^s h^t)^{e_2}\} \\ &= \{g^{e_1x_1+e_2s}h^{e_1y_1+e_2t}, \cdots, g^{e_1x_n+e_2s}h^{e_1y_n+e_2t}\}. \end{aligned}$$

Where $e_1 = H(L_{PK}, TK, 1)$, $e_2 = H(L_{PK}, TK, 2)$, moreover, we assume $e_i \neq 0$ for $i = 1, 2$, except for negligible probability ϵ_0. According to the condition that $\mathcal{A}$ signed τ with PK_κ, then we get $\mathcal{A}$ knows the corresponding $SK_\kappa = z$, except for negligible probability ϵ_1, under the unforgeability of ring signature, that is:

$$\begin{aligned} g^{e_1x_\kappa+e_2s}h^{e_1y_\kappa+e_2t} &= (g^{e_1}h^{e_2})^{e_1^{-1}(e_1x_\kappa+e_2s)}h^{e_1y_\kappa+e_2t-e_1^{-1}e_2(e_1x_\kappa+e_2s)} \\ &= (g^{e_1}h^{e_2})^z. \end{aligned}$$

In the rest of the proof, we prove that $x_\kappa = t$ and $s = y_\kappa = 0$. First, if $e_1y_\kappa + e_2t - e_1^{-1}e_2(e_1x_\kappa + e_2s) \neq 0$, following the similar arguments in Theorem 9, we know that $\mathcal{A}$ gets a non-trivial relationship between g and h, this happens with negligible probability ϵ_2 according to the hardness of discrete logarithm problems. Then we get $e_1y_\kappa + e_2t - e_1^{-1}e_2(e_1x_\kappa + e_2s) = 0$ with nonnegligible probability $\delta - \epsilon_0 - \epsilon_1 - \epsilon_2$. Then we have $e_1^2y_\kappa + e_1e_2(t - x_\kappa) + e_2^2s = 0$, and e_2 is exactly the solution for equation $sx^2 + e_1(t - x_\kappa)x + e_1^2y_\kappa = 0$ in $\mathbb{Z}_q$ after $e_1 = H(L_{PK}, TK, 1)$ was generated, which has at most two solutions when $s(t - x_\kappa) \neq 0$ and q is a prime. This means e_2 is pre-determined (1 out of 2) before $\mathcal{A}$ runs the hash function (random oracle), this also happens with negligible probability ϵ_3, then we get $x_\kappa = t$ and $s = y_\kappa = 0$, which means the equation degenerated to zero. Then we have $PK_\kappa^y = g^{ty} = h^t = TK$, then $\mathsf{Trace}(\sigma, y) = \kappa$, which contradicted with the assumptions before, then we finish the traceability proof of sTLRS. □

5 Modification

5.1 Construction

In this section, we give a modification of sTLRS to achieve security against malicious auditors, which means that even for adversary $\mathcal{A}$ with possession of

the trapdoor, he still cannot double spend, slander honest users, forge sTLRS signatures or escape from audit. Together with the traceable range proofs and traceable scheme for long-term addresses[14], we can finally finish a new construction of fully auditable privacy-preserving blockchains, with better efficiency and smaller size. In the following we give the detailed description of the modified sTLRS, which is named by sTLRS':

- $\mathsf{Par} \leftarrow \mathsf{Setup'}(\lambda)$:
  1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and a generator $g_1 \in \mathbb{G}$, the auditor generates $y \in \mathbb{Z}_q$ as the trapdoor, computes $h = g_1^y$, system computes $g_2 = H_p(g_1, h)$ (use hash to point), system outputs $(\mathbb{G}, q, g_1, g_2, h)$ as the public parameters.
- $(PK, SK) \leftarrow \mathsf{KeyGen'}(\mathsf{Par})$:
  1. According to the public parameters $(\mathbb{G}, q, g_1, g_2, h)$, user Alice samples $x \in \mathbb{Z}_q$ as her secret key, then computes $PK = g_1^x$;
  2. Alice outputs $PK = g_1^x$, and retains $SK = x$.
- $\sigma \leftarrow \mathsf{Sign'}(SK_\kappa, \mu, L_{PK})$:
  1. For a message μ, Alice chooses another $n-1$ users, together with her own public key, to generate a list of public keys $L_{PK} = \{PK_1, \cdots, PK_n\}$, where Alice's $PK = PK_\kappa \in L_{PK}, \kappa \in \{1, \cdots, n\}$;
  2. Alice outputs $I = g_2^{x_\kappa}, TK = h^{x_\kappa}$, then computes $e_1 = H(L_{PK}, I, TK, 1)$, $e_2 = H(L_{PK}, I, TK, 2)$ and $e_3 = H(L_{PK}, I, TK, 3)$;
  3. Alice computes and outputs

$$L_{RK} = \{PK_1^{e_1} \cdot I^{e_2} \cdot TK^{e_3}, \cdots, PK_n^{e_1} \cdot I^{e_2} \cdot TK^{e_3}\}$$

$$= \{g_1^{e_1 x_1} g_2^{e_2 x_\kappa} h^{e_3 x_\kappa}, \cdots, g_1^{e_1 x_n} g_2^{e_2 x_\kappa} h^{e_3 x_\kappa}\};$$

  4. Alice runs classic ring signature $\tau \leftarrow \mathsf{Rsign}(SK, \mu, L_{RK}, I, TK)$ using L_{RK} and $SK = x_\kappa$, outputs τ ($g_1^{e_1} g_2^{e_2} h^{e_3}$ as the generator);
  5. Alice outputs $\sigma = (\tau, \mu, L_{PK}, I, TK)$.
- $1/0 \leftarrow \mathsf{Verify'}(\tau, \mu, L_{PK}, I, TK)$:
  1. Verifier computes $e_1^* = H(L_{PK}, I, TK, 1)$, $e_2^* = H(L_{PK}, I, TK, 2)$ and $e_3^* = H(L_{PK}, I, TK, 3)$;
  2. Verifier computes $L_{RK}^* = \{PK_1^{e_1^*} \cdot I^{e_2^*} \cdot TK^{e_3^*}, \cdots, PK_n^{e_1^*} \cdot I^{e_2^*} \cdot TK^{e_3^*}\}$;
  3. Verifier checks the validity of ring signature τ ($g_1^{e_1^*} g_2^{e_2^*} h^{e_3^*}$ as the generator);
  4. If all passed then outputs 1, otherwise outputs 0.
- *linked/unlinked* $\leftarrow \mathsf{Link'}(\sigma, \sigma')$:
  1. For two valid sTLRS' signatures $\sigma = (\tau, \mu, L_{PK}, I, TK)$ and $\sigma' = (\tau', \mu', L'_{PK}, I', TK')$, if $I = I'$ then verifier outputs *linked*, otherwise outputs *unlinked*.
- $\kappa^* / \perp \leftarrow \mathsf{Trace'}(\sigma, y)$:
  1. For $\sigma = (\tau, \mu, L_{PK}, I, TK)$, the auditor extracts $PK_1, \cdots, PK_n$ from L_{PK}, computes PK_i^y for $i = 1, \cdots, n$, outputs the smallest $\kappa^* \in \{1, \cdots, n\}$ such that $TK = PK_{\kappa^*}^y$ as the trace result, otherwise outputs $\perp$.

5.2 Correctness

Theorem 13 (Correctness) *For an honest user Alice in sTLRS', she can complete the traceable and linkable ring signature successfully, and the behavior of double signing (double spending) will be detected while the identity of Alice remaining anonymous. Moreover, the auditor can trace her identity correctly.*

Proof. In sTLRS', for Alice's public key $PK = PK_\kappa = g_1^{x_\kappa}$, then Alice will output $I = g_2^{x_\kappa}$ and $TK = h^{x_\kappa}$ with $L_{RK} = \{g_1^{e_1 x_1} g_2^{e_2 x_\kappa} h^{e_3 x_\kappa}, \cdots, g_1^{e_1 x_n} g_2^{e_2 x_\kappa} h^{e_3 x_\kappa}\}$. Since $g_1^{e_1 x_\kappa} g_2^{e_2 x_\kappa} h^{e_3 x_\kappa} = (g_1^{e_1} g_2^{e_2} h^{e_3})^{x_\kappa}$, then Alice can use $SK = x_\kappa$ to generate the ring signature τ using $g_1^{e_1} g_2^{e_2} h^{e_3}$ as the generator.

When double signing occurs, we know from the linkability of sTLRS' that Alice must have used $I = g_2^{x_\kappa}$ for twice (similar to sTLRS), then the verifier can detect that double signing occurs and outputs *linked*, at the same time, anyone (except for the auditor) cannot learn any information about the identity of signer by the anonymity of sTLRS' (similar to sTLRS).

For the auditor, he can compute $PK_\kappa^y = g_1^{y x_\kappa} = h^{x_\kappa} = TK$ and then outputs $\mathsf{Trace}(\sigma, y) = \kappa$ correctly. □

5.3 Security

Following the same direction of Theorem 8,9,10,12, we can prove the anonymity, linkability, nonslanderability, unforgeability, traceability of sTLRS', for any PPT adversary $\mathcal{A}$ with possession of the trapdoor, detailed proofs will be given in the full version of this paper.

6 Traceable and Linkable Multi-ring Signature

6.1 Construction

In this section, we give the construction of TLMS by usage of key-image generation with randomized base $h = H_p(PK)$, which is similar in the Monero system. Moreover, we also need to modify the generation algorithm of tracing keys TKs to ensure the security of the system. Actually, in the multi-ring version (m-ring signature), the linkability works for the first $m-1$ rings and the traceability works for the last ring. Here we give an example of 2-ring version, which can be used in the one-to-many transactions in Monero-type cryptocurrency. We choose the (extended) AOS ring signature as component.

- $\mathsf{Par} \leftarrow \mathsf{TLMS.Setup}(\lambda)$:
  1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, the auditor generates $z \in \mathbb{Z}_q$ as the trapdoor, computes $h = g^z$, system outputs $(\mathbb{G}, q, g, h)$ as the public parameters.
- $(\mathsf{PK},\mathsf{SK}) \leftarrow \mathsf{TLMS.KeyGen}(\mathsf{Par})$:
  1. According to the public parameters $(\mathbb{G}, q, g, h)$, user Alice samples $x, y \in \mathbb{Z}_q$ as her secret key;

2. Alice outputs $(\mathsf{PK}, \mathsf{PK}') = (g^x, g^y)$, and retains $(\mathsf{SK}, \mathsf{SK}') = (x, y)$.

– $\sigma \leftarrow \mathsf{TLMS.Sign}(\mathsf{SK}, \mathsf{SK}', \mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}})$:
1. For a message μ, Alice chooses another $n-1$ users, together with her own public key, to generate two lists of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\}$ and $L'_{\mathsf{PK}} = \{\mathsf{PK}'_1, \cdots, \mathsf{PK}'_n\}$, where Alice's $\mathsf{PK} = \mathsf{PK}_\pi \in L_{\mathsf{PK}}, \mathsf{PK}' = \mathsf{PK}'_\pi \in L'_{\mathsf{PK}}, \pi \in \{1, \cdots, n\}$, which means the position of Alice's public key in each ring is same;
2. Alice computes $h_i = H_p(\mathsf{PK}_i)$ for $i = 1, \cdots, n$, then computes the key-image $I = h_\pi^{x_\pi}, \mathsf{TK} = h^{y_\pi}$, then computes $e_k = H(L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK}, \mu, k)$ for $k = 1, 2$;
3. Alice computes

$$L_1 = \{\mathsf{PK}_1 \cdot I^{e_1}, \cdots, \mathsf{PK}_n \cdot I^{e_1}\} = \{g^{x_1} h_\pi^{e_1 x_\pi}, \cdots, g^{x_n} h_\pi^{e_1 x_\pi}\},$$

$$L_2 = \{\mathsf{PK}'_1 \cdot \mathsf{TK}^{e_2}, \cdots, \mathsf{PK}'_n \cdot \mathsf{TK}^{e_2}\} = \{g^{y_1} h^{e_2 y_\pi}, \cdots, g^{y_n} h^{e_2 y_\pi}\};$$

4. Alice runs the position-preserving multi-ring signature $\tau \leftarrow \mathsf{Rsign}(\mathsf{SK}, \mathsf{SK}', \mu, L_1, L_2, I, \mathsf{TK})$ with $(\mathsf{SK}, \mathsf{SK}') = (x, y)$, outputs τ (in L_1, we use generator $gh_i^{e_1}$ in the i-th position; in L_2, we use generator gh^{e_2} in all positions);
5. Alice outputs $\sigma = (\tau, \mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK})$.

– $1/0 \leftarrow \mathsf{TLMS.Verify}(\tau, \mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK})$:
1. Verifier computes $h_i = H_p(\mathsf{PK}_i)$ for $i = 1, \cdots, n$, then computes $e_1^* = H(L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK}, \mu, 1)$ and $e_2^* = H(L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK}, \mu, 2)$;
2. Verifier computes

$$L_1 = \{\mathsf{PK}_1 \cdot I^{e_1^*}, \cdots, \mathsf{PK}_n \cdot I^{e_1^*}\}, L_2 = \{\mathsf{PK}'_1 \cdot \mathsf{TK}^{e_2^*}, \cdots, \mathsf{PK}'_n \cdot \mathsf{TK}^{e_2^*}\};$$

3. Verifier checks the validity of position-preserving multi-ring signature τ (in L_1, verifier uses generator $gh_i^{e_1^*}$ in the i-th position; in L_2, he uses generator $gh^{e_2^*}$ in all positions);
4. If all passed then outputs 1, otherwise outputs 0.

– *linked/unlinked* $\leftarrow \mathsf{TLMS.Link}(\sigma, \sigma')$:
1. For two valid TLMS signatures $\sigma_1 = (\tau_1, \mu_1, L_{\mathsf{PK}}^{(1)}, L_{\mathsf{PK}}^{\prime(1)}, I_1, \mathsf{TK}_1)$ and $\sigma_2 = (\tau_2, \mu_2, L_{\mathsf{PK}}^{(2)}, L_{\mathsf{PK}}^{\prime(2)}, I_2, \mathsf{TK}_2)$, if $I_1 = I_2$ then verifier outputs *linked*, otherwise outputs *unlinked*.

– $\kappa^* / \perp \leftarrow \mathsf{TLMS.Trace}(\sigma, z)$:
1. For $\sigma = (\tau, \mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, I, \mathsf{TK})$, the auditor extracts $\mathsf{PK}'_1, \cdots, \mathsf{PK}'_n$ from L'_{PK}, computes $\mathsf{PK}_i'^z$ for $i = 1, \cdots, n$, outputs the smallest $\kappa^* \in \{1, \cdots, n\}$ such that $\mathsf{TK} = \mathsf{PK}_{\kappa^*}'^z$ as the trace result, otherwise outputs $\perp$.

6.2 Applications in Monero-type Cryptocurrency

In Monero system, every UTXO has its public-private key pair $(\mathsf{PK} = g^s, \mathsf{SK} = s)$ and the corresponding value commitment $c = g^x h^a$ (or $c = h^x g^a$ in Bulletproofs), where c is Pedersen commitment[22], a is the hidden value and x is the blinding element. In a transaction, the initiator Alice chooses $m-1$ hiding UTXOs: $\{(\mathsf{PK}_i, c_i = g^{x_i} h^{a_i})\}_{i=1,\cdots,m-1}$, along with her input UTXO $(\mathsf{PK}_A = g^s, c_A =$

$g^{x_A}h^{a_A}$), to generate a set of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_A, \mathsf{PK}_1, \cdots, \mathsf{PK}_{m-1}\}$ (randomized order), Alice also generates the output UTXO $(\mathsf{PK}_B, c_B = g^{x_B}h^{a_B})$ (the generation algorithm of PK_B is in the Appendix), where the input value equals the output value $a_A = a_B$. Then Alice computes another ring of commitments (same order as in L_{PK}):

$$L'_{\mathsf{PK}} = \{c_A c_B^{-1}, c_1 c_B^{-1}, \cdots, c_{m-1} c_B^{-1}\}$$

$$= \{g^{x_A - x_B}, g^{x_1 - x_B}h^{a_1 - a_B}, \cdots, g^{x_{m-1} - x_B}h^{a_{m-1} - a_B}\}.$$

Alice uses linkable 2-ring signature to sign the transaction by L_{PK} and L'_{PK}, with the same position of signing key in each ring, we call it the position-preserving linkable multi-ring signature. In this paper, TLMS is suited in the Monero-type cryptocurrency.

7 Conclusion

In this paper, we give a simpler and modular construction of traceable and linkable ring signature scheme (sTLRS) by modifying the key generation algorithm and removing the additional one-time signature and zero-knowledge proofs, which reduces the size of public key, shortens the computation time, realizes the regulatory function for signers' identities, and can prevent the adversary from double spending, escaping from audit, slandering users or forging signatures. Moreover we give a traceable and linkable multi-ring signature (TRMS) to support Monero-type cryptocurrency. Our work is a new approach to construct auditable privacy-preserving blockchains and cryptocurrencies, and is a potential replacement for Monero-type blockchains.

Future Works In the future, we need to study and improve in the following aspects:

1. Study new method to construct traceable range proof with less verification time and smaller size;
2. Study post-quantum ring signatures and range proofs, such as lattice-based, code-based, multi-variant-based and isogen-based schemes to prepare for the future applications and replacement in the era of quantum computing.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 415–432. Springer (2002)
2. Au, M.H., Chow, S.S., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: European Public Key Infrastructure Workshop. pp. 101–115. Springer (2006)

3. Back, A.: Ring signature efficiency. Bitcointalk (accessed 1 May 2015) (2015), https://bitcointalk.org/index.php
4. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Theory of Cryptography Conference. pp. 60–79. Springer (2006)
5. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018)
6. Buterin, V.: A next-generation smart contract and decentralized application platform (2014), https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf
7. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: International Colloquium on Automata, Languages, and Programming. pp. 423–434. Springer (2007)
8. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 609–626. Springer (2004)
9. Duffield, E., Diaz, D.: Dash: A privacycentric cryptocurrency. GitHub (2015), https://github.com/dashpay/dash/wiki/Whitepaper
10. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
11. Goodell, B., Noether, S., RandomRun: Compact linkable ring signatures and applications. Cryptology ePrint Archive, Report 2019/654 (2019), https://eprint.iacr.org/2019/654
12. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 253–280. Springer (2015)
13. Jedusor, T.E.: Mimblewimble (2016), http://mimblewimble.cash/20160719-OriginalWhitePaper.txt
14. Li, W., Chen, L., Lai, X., Zhang, X., Xin, J.: Fully regulatable privacy-preserving blockchains against malicious regulators. Cryptology ePrint Archive, Report 2019/925 (2019), https://eprint.iacr.org/2019/925
15. Li, Y., Yang, G., Susilo, W., Yu, Y., Au, M.H., Liu, D.: Traceable monero: Anonymous cryptocurrency with enhanced accountability. IEEE Transactions on Dependable and Secure Computing (2019)
16. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Transactions on Knowledge and Data Engineering **26**(1), 157–165 (2013)
17. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
18. Maxwell, G.: Confidential transactions (2015), https://people.xiph.org/~greg/confidential_values.txt
19. Maxwell, G., Poelstra, A.: Borromean ring signatures (2015), https://raw.githubusercontent.com/Blockstream/borromean_paper/master/borromean_draft_0.01_34241bb.pdf
20. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008), https://git.dhimmel.com/bitcoin-whitepaper/
21. Noether, S., Mackenzie, A., et al.: Ring confidential transactions. Ledger **1**, 1–18 (2016)

22. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual International Cryptology Conference. pp. 129–140. Springer (1991)
23. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–565. Springer (2001)
24. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
25. Sun, S.F., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security. pp. 456–474. Springer (2017)
26. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: International Conference on Information Security Practice and Experience. pp. 48–60. Springer (2005)
27. Van Saberhagen, N.: Cryptonote v 2.0 (2013), `https://cryptonote.org/whitepaper.pdf`
28. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. The Computer Journal **56**(4), 407–421 (2013)
29. Yuen, T.H., Sun, S.f., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security (2019)

A Remaining Preliminaries

A.1 AOS Ring Signature

We give the introduction of AOS ring signature[1] in the following: here we introduce the generalized AOS ring signature for the generator in each position is different (g_i is the generator in the i-th position for $i = 1, \cdots, n$).

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator $g_1, \cdots, g_n$ as the public parameters.
- $(\mathsf{PK}_\pi, \mathsf{SK}_\pi) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$: according to the public parameters, user P_π samples $x \in \mathbb{Z}_q^*$ uniformly at random, computes g_π^x and sets $(\mathsf{PK}_\pi, \mathsf{SK}_\pi) = (g_\pi^x, x)$.
- $\sigma \leftarrow \mathsf{Rsign}(\mathsf{SK}_\pi, \mu, L_{\mathsf{PK}})$: when user P_π generates a ring signature for message μ, he chooses another $n-1$ users' public keys, together with his own PK_π to obtain a set of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\} = \{g_1^{x_1}, \cdots, g_n^{x_n}\}$, where $\mathsf{PK}_\pi \in L_{\mathsf{PK}}$ and $\pi \in \{1, \cdots, n\}$, then he does as follows:
  1. P_π samples $r_\pi \in \mathbb{Z}_q^*$ uniformly at random, then computes $c_{\pi+1} = H(g_\pi^{r_\pi}, L_{\mathsf{PK}}, \mu)$;
  2. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, P_π samples $z_i \in \mathbb{Z}_q^*$ uniformly and computes $c_{i+1} = H(g_i^{z_i}/(\mathsf{PK}_i)^{c_i}, L_{\mathsf{PK}}, \mu)$;
  3. P_π computes $z_\pi = r_\pi + xc_\pi$;
  4. Output the ring signature $\sigma = (c_1; z_1, \cdots, z_n)$.

– $1/0 \leftarrow \mathsf{Verify}(\mu, \sigma, L_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, \sigma)$, for $i = 1, \cdots, n$ the verifier computes

$$c_{i+1}^* = H(g_i^{z_i}/(\mathsf{PK}_i)^{c_i^*}, L_{\mathsf{PK}}, \mu)$$

where $c_1 = c_1^*$, then checks $c_1 \stackrel{?}{=} c_{n+1}^*$, if all passed then outputs 1, otherwise outputs 0.

A.2 AOS 2-ring Signature

We give the introduction of AOS 2-ring signature as the key component of TLMS, we assume the generators in different rings and different positions are different (g_i is the generator in the i-th position for $i = 1, \cdots, n$ in L_1, and h_i is the generator in the i-th position for $i = 1, \cdots, n$ in L_2).

– $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator $g_1, \cdots, g_n, h_1, \cdots, h_n$ as the public parameters.
– $(\mathsf{PK}_\pi, \mathsf{SK}_\pi) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$: according to the public parameters, user P_π samples $x, y \in \mathbb{Z}_q^*$ uniformly at random, computes g_π^x, h_π^y and sets $(\mathsf{PK}_\pi, \mathsf{PK}'_\pi, \mathsf{SK}_\pi, \mathsf{SK}'_\pi) = (g_\pi^x, h_\pi^y, x, y)$.
– $\sigma \leftarrow \mathsf{Rsign}(\mathsf{SK}_\pi, \mathsf{SK}'_\pi, \mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}})$:
  1. For a message μ, Alice chooses another $n-1$ users, together with her own public keys, to generate two lists of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\}$ and $L'_{\mathsf{PK}} = \{\mathsf{PK}'_1, \cdots, \mathsf{PK}'_n\}$, where Alice's $\mathsf{PK} = \mathsf{PK}_\pi \in L_{\mathsf{PK}}, \mathsf{PK}' = \mathsf{PK}'_\pi \in L'_{\mathsf{PK}}, \pi \in \{1, \cdots, n\}$, which means the position of Alice's public key in each ring is same
  2. P_π samples $r_\pi, s_\pi \in \mathbb{Z}_q^*$ uniformly at random, then computes $c_{\pi+1} = H(g_\pi^{r_\pi}, h_{\pi+1}^{s_\pi}, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, \mu)$;
  3. For $i = \pi + 1, \cdots, n, 1, \cdots, \pi - 1$, P_π samples $z_i, z'_i \in \mathbb{Z}_q^*$ uniformly and computes $c_{i+1} = H(g_i^{z_i}/(\mathsf{PK}_i)^{c_i}, h_i^{z'_i}/(\mathsf{PK}'_i)^{c_i}, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, \mu)$;
  4. P_π computes $z_\pi = r_\pi + xc_\pi$ and $z'_\pi = s_\pi + yc_\pi$;
  5. P_π outputs the ring signature $\sigma = (c_1; z_1, \cdots, z_n; z'_1, \cdots, z'_n)$.
– $1/0 \leftarrow \mathsf{Verify}(\mu, \sigma, L_{\mathsf{PK}}, L'_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, \sigma)$, for $i = 1, \cdots, n$ the verifier computes

$$c_{i+1}^* = H(g_i^{z_i}/(\mathsf{PK}_i)^{c_i^*}, h_i^{z'_i}/(\mathsf{PK}'_i)^{c_i^*}, L_{\mathsf{PK}}, L'_{\mathsf{PK}}, \mu)$$

where $c_1 = c_1^*$, then checks $c_1 \stackrel{?}{=} c_{n+1}^*$, if all passed then outputs 1, otherwise outputs 0.