Signer Revocability for Threshold Ring Signatures

Da Teng[1,2,3] and Yanqing Yao[1,2,3*]

[1*]State Key Laboratory of Software Development Environment, Beihang University, Beijing, 100191, China.
[2]State Key Laboratory of Cryptology,Beijing, 100878, China.
[3]Key Laboratory of Aerospace Network Security, Ministry of Industry and Information Technology, School of Cyber Science and Technology, Beihang University, Beijing, 100191, China.

Abstract

$\boldsymbol{t}$-out-of-$\boldsymbol{n}$ threshold ring signature (TRS) is a type of anonymous signature designed for $\boldsymbol{t}$ signers to jointly sign a message while hiding their identities among $\boldsymbol{n}$ parties that include themselves. However, can TRS address those needs if one of the signers wants to revoke his signature or, additively, sign separately later? Can non-signers be revoked without compromising anonymity? Previous research has only discussed opposing situations. The present study introduces a novel property for TRS-revocability- addressing the need for improved flexibility and privacy security in TRS. Our proposed revocable threshold ring signature (RTRS) scheme is innovative in several ways: (1) It allows a signer to non-interactively revoke their identity and update the signature from $\boldsymbol{t}$-out-of-$\boldsymbol{n}$ to $\boldsymbol{t-1}$-out-of-$\boldsymbol{n}$; (2) It is possible to reduce the ring size and clip non-signers along with revoked signers while maintaining the anonymity level. We analyze and define the boundaries for these operations and implement and evaluate our structure. With a sufficiently large ring size, we can optimize the signature size, resulting in better signing performance as compared to the extensible signature scheme.

Keywords: Threshold ring signatures, Revocability, Anonymity, Electronic voting

1 Introduction

Threshold ring signatures (TRS) [1] are an extension of ring signature (RS) schemes and a type of digital signature that certifies t signers, among a group of n participants, jointly signed a specific message, without revealing which are the actual signers. Similar to ring signatures, this group of participants is referred to as a "ring", and any verifier cannot distinguish the members of the ring who are signers from those who are not. Moreover, signers do not need to inform non-signers who are used as the anonymous set when generating signatures. The anonymity of (threshold) ring signatures makes them very useful in some privacy-demanding scenarios, such as e-voting or elections, where the results need to be collected without revealing the identity of supporters or excluding the effect of supporters' positions on the proposals. Existing anonymous voting mechanisms lack provisions for pre-voting or revoking votes, allowing some voters who have already cast their ballots to revoke them before the final tally. Allowing participants to regret can help ensure fair voting, encourage thoughtful decision-making, and avoid regrettable decisions made on impulse.

In psychology, there exists a phenomenon known as "buyer's remorse", commonly used to

describe post-purchase regrets. Similarly, buyer's remorse for voters (or voters' remorse) specifically refers to the situation where individuals feel regret after an election or voting process. Voters may experience unease regarding their chosen candidates or voting decisions after casting their ballots. This regret may stem from a reassessment of a candidate's performance, policy positions, or other factors. This psychological phenomenon is widespread among populations, with its roots in decision-making processes, individual values, and societal influences, among other factors. Therefore, considering these factors in the design of voting systems and allowing voters the opportunity for reconsideration is meaningful for the fairness of elections or voting processes.

In the context of TRS, what does a voter's "regret" mean? In a $(t, n)-$TRS signature, if a signer, i.e., a participating voter, decides to withdraw his vote, he must revoke his signer identity in the signature, which can be achieved by downgrading the identity to a non-signer in the ring. This results in a reduction of the signature threshold from t to $t-1$, thus decreasing the level of anonymity of the signers in the ring from t/n to $(t-\alpha)/n$, where α is the number of revoked signers. Moreover, this provides an opportunity to improve storage efficiency by appropriately reducing anonymity. On a threshold ring signature that has experienced partial signer revocation, the elimination of certain confounding options, i.e., non-signer within the ring, through an operation known as "clipping", is effective in conserving the actual storage space of the ring signature. This efficiency stems from the existing construction of ring signatures, where the size of the signature is closely correlated with the scale of the ring. Still, given the initial anonymity of the co-signers who generated the signature, the anonymity of the signature cannot be reduced below the consensus threshold of t/n, regardless of how many rounds of revocation have taken place. In electronic voting, this clipping process offers an additional level of flexibility by effectively reducing the actual storage requirements of voting information without compromising the anonymity of the voters. This proves particularly advantageous in large-scale electronic voting scenarios with heightened confidentiality requirements.

As shown in Figure 1, our voting process supports the revocation of votes. Voters can retract their votes after casting their preliminary ballots before the final tally is counted, allowing for dynamic adjustments. Further, we consider how to achieve this. Even after revocation, the revoked voters remain members of the ring, which means that in the view of any third-party verifier, they may still be signers. Can we let the revoked signers directly exit the signature ring? The answer is no. In a (T)RS, the public key set that form the ring is public, which means that an attacker can identify who the former signers were based on the difference between the ring public keys used before and after modification. Although the votes have been revoked, the fact that some specific voters changed their voting intention may still affect the fairness of the election results. For signatures with revoked ring members, trimming a portion of non-signing ring members can help reduce the size of the signatures, corresponding to the dynamic adjustment in our voting application. This is because, in most (T)RS schemes, the signature size is positively correlated with the size of the ring. Our proposed scheme is also based on modifying and expanding the construction of ring signatures with $\mathcal{O}(n)$ space complexity. Therefore, a more reasonable way to reduce the ring size is to randomly clip members of the non-signing set in the ring, including participants who have been revoked and demoted to non-signers. Through this approach, although third-party attackers can still identify the pruned ring members by comparison, they cannot confirm whether they have ever signed the vote. Consequently, dynamic adjustments effectively reduce the storage space requirements for the election results.

Fig. 1: Dynamic electronic voting application workflow

1.1 Related Work

There were some similar notions in early research regarding revocability, but they were not equivalent to the "revocability" described in this work.

Partially revocable signature. As is often the practice case, exposing only part of the signatures of a given set of messages may be desired while ensuring that the remaining signatures are only valid on a subset of the messages. A construction of RSA-based signature is proposed by Nojima et al. [2], which hides part of messages for sensitive data release or other particular scenarios without random oracle. Similarly, Brzuska et al. [3] introduced the concept of a tree-structured signature, which allows for the revocation of part of the signature through subtree revision while maintaining the validity of other components. This approach has gained attention and has been applied in several blockchain-related research efforts. Additionally, Camenisch et al. [4] proposed an unlinkable editable signature scheme that allows for the partial display of a signature and was used to build the first effective UC-secure anonymous credential system. Sanders [5] also proposed a constant size modifiable signature scheme that supports editing a message signature to make it valid only on a subset of its signature messages.

Revocability of RS. A separable TRS scheme was proposed by Liu et al. [6], which defines separability as the ability to use various kinds of public keys in the same signature scheme, such as RSA and DL. It should be noted that separability, while not involving any changes to the signer set or anonymity set of TRS signatures, enables separate processing by different signers and has inspirational significance for subsequent research. Coincidentally, Liu et al. [7] later formally proposed a revocability property of ring signatures. They introduced a trusted authority and defined revocability as allowing the authority to de-anonymize signers and reveal their true identities. Clearly, this is not the same as the revocability described in this paper, and the goal of de-anonymization is inconsistent with the goal of maintaining anonymity in our proposal.

Previous research has defined the property of changing the threshold size of signers in TRS. Okamoto et al. [8] proposed and defined "flexibility" to update generated signatures to increase the number of signers. Similar to our scheme, both reuse the original signature and do not require interaction from other signers to make changes to the signature. However, they did not consider the case of reducing the number of signers without interaction, which is a valuable property in practice, as described earlier. The signer revocation proposed in this paper complements the definition of "flexibility". In response to the issue of fixed potential signers once generated, Aranha et al. [9] further defined the notion of "extendability" based on flexibility, in addition to proposing a TRS scheme that not only satisfies threshold extendability but also allows for the expansion of the potential signer set. Moreover, they proved the indistinguishability of the execution sequence and rounds for the two functions of the proposed extendability. However, they did not consider the opposite of extendability, i.e., revocability, which includes the revocation of signers for the threshold and the reduction of the potential signer set while maintaining anonymity, as well as the indistinguishability of the revocation order. Gennaro et al. [10] pointed out a weakness in the work of [9]: it only guarantees anonymity against attackers who just observe the final signature and cannot access the "full evolution" of ETRS. Our proposed revocable signature inherently addresses this deficiency, meeting the slightly stronger anonymity requirement proposed by [10]. Regardless of "flexibility" or "extendability", the proposed scheme in this paper supplements the opposite properties to their definitions, which have not been systematically defined or implemented in previous research.

1.2 Contributions

The following are our contributions in this work:

- We introduce a novel concept called revocability of TRS. Our revocability has an opposite property to the extendability proposed in [9] and is distinct from the commonly understood concept of "signature revocation".
- We define the first sub-property of revocability, called clippability, which allows the signer of a (T)RS to reduce the anonymity set of a given signature. We provide the first construction of the clippable ring signature (CRS).
- We define the second sub-property of revocability, called splittability, which allows the signer of a TRS to exit the set of signers for a given signature while preserving the anonymity set. We present the first splittable threshold ring signature (STRS) construction:

  - By employing improved sum argument to establish threshold relationships, we convert the above ring signature construction to a TRS construction.
  - Based on this TRS, we provide the STRS construction.

- Combining the above two sub-properties, we first construct a revocable threshold ring signature (RTRS, as Figure 2 shows). It is important to note that through analysis and control of the trimming boundary, the level of anonymity for the signers is maintained at a minimum of the same level as the original.
- Security definitions and proofs are presented for the proposed scheme, along with an evaluation and comparison of signature generation time and signature size for our implementation. The results demonstrate that, our proposed scheme can complete signature generation in less time and effectively save storage space when the ring size is sufficiently large.

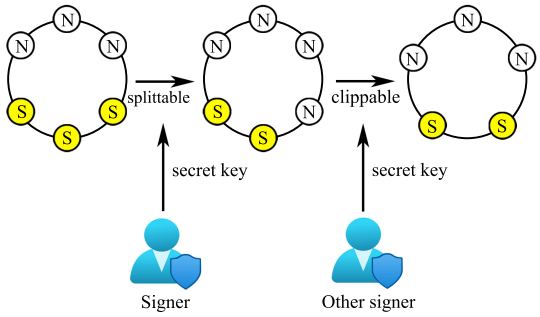


Fig. 2: Definition of revocability

1.3 Organization

The following sections of this paper are structured as follows. Sec. 2 provides some preliminaries, while in Sec. 3, the security model is defined. Then we offer our proposed CRS, STRS, and RTRS in Sec. 4-6, respectively. We implement and evaluate our schemes in Sec. 7. We conclude in Sec. 8.

2 Preliminaries

2.1 (Threshold) Ring Signature

Let us revisit the definition of a ring signature:

Ring signature [11]. A tuple of polynomial-time algorithms **RS** = (Setup, KeyGen, Sign, Verify) forms a ring signature scheme:

- Setup(1^λ) $\rightarrow$ pp. The public parameters, denoted as pp, are outputted and implicitly serve as inputs to the subsequent algorithms, given a security parameter λ.
- KeyGen() $\rightarrow (sk, pk)$. Produce a secret-public key pair.
- Sign$(m, \{pk_i\}_{i\in\mathcal{R}}, sk) \rightarrow \sigma$. Given a message m to be signed, a ring of public keys $\{pk_i\}_{i\in\mathcal{R}}$ with index set $\mathcal{R}$, and the signer's secret key sk, this algorithm produces a ring signature σ.
- Verify$(m, \{pk_i\}_{i\in\mathcal{R}}, \sigma) \rightarrow 0/1$. Given a message m, a ring of public keys $\{pk_i\}_{i\in\mathcal{R}}$, and a signature σ, this algorithm outputs a bit $b \in \{0,1\}$, where $b = 1$ indicates a valid signature and $b = 0$ indicates an invalid signature.

Dualring ring signature. Dualring is a generic construction for ring signatures, as described in [12]. Schnorr signature is one possible instantiation of Dualring, which has been proven to be secure in the random oracle model with respect to unforgeability and anonymity. Next, we introduce the signing and verifying part of this DL-based Dualring instance $\texttt{Dualring}_{EC}$, shown in Algorithm 1, while some details are omitted.

Note that this work refers to the original Dualring signature structure but not the optimized logarithmic size version.

The security of DualRing is defined by Unforgeability w.r.t. Insider Corruption and Anonymity against full key exposure.

(**Unforgeability w.r.t Insider Corruption**). For any polynomial-time adversary $\mathcal{A}$, a ring signature is considered unforgeable if there exists a polynomial related to integers q_k in λ, where λ is the security parameter:

$$\Pr\left[\begin{array}{c|c} 1 \leftarrow Verify(m^*, & param \leftarrow Setup(\lambda), \\ \{pk_i^*\}_{i\in\mathcal{R}}, \sigma^*), & for\ i \in [1, q_k]: \\ \{pk_i^*\}_{i\in\mathcal{R}} \subseteq S\backslash C, & (pk_i', sk_i') \leftarrow KeyGen(), \\ (m^*, \{pk_i^*\}_{i\in\mathcal{R}}, \cdot) & S := \{pk_i'\}_{i=1}^{q_k}, \\ was\ not\ the & (m^*, \{pk_i^*\}_{i\in\mathcal{R}}, \sigma^*) \leftarrow \\ input\ of\ \mathcal{OS} & \mathcal{A}^{\mathcal{OC},\mathcal{OS}}(param, S) \end{array}\right] \leq (\lambda).$$

The two oracle definitions involved are as follows:

Algorithm 1: $Dualring_{EC}$. Sign & Verify

Sign-Input : param, $m, \{pk_i\}_{i\in\mathcal{R}}, sk_j$
Sign-Output: σ
random r, c_i for all $i \neq j$;
$R = g^r \cdot \prod_{i\neq j} pk_i^{c_i}$;
$c = H(m, \{pk_i\}_{i\in\mathcal{R}}, R)$;
$c_j = c - \sum_{i\neq j} c_i$;
$z = r - sk_j \cdot c_j$;
return $\sigma = (c_1, ..., c_n, z)$;
Verify-Input : param, m, $\{pk_i\}_{i\in\mathcal{R}}, \sigma$
Verify-Output: 0 or 1
parse $\sigma = (c_1, ..., c_n, z)$;
$R = g^z \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i}$;
$c = \sum_{i=1}^{n} c_i$;
if $c \neq H(m, \{pk_i\}_{i\in\mathcal{R}}, R)$ **then**
  return 0;
return 1.

- $\mathcal{OC}(i)$: Corruption Oracle, outputs sk'_i. Let the set of corruption queries for $\mathcal{OC}(i)$ be denoted as C.
- $\mathcal{OS}(m, \{pk_i\}_{i\in\mathcal{R}}, j)$: Signature Oracle, where j is the index of the signer, outputs the signature $\sigma \leftarrow (m, \{pk_i\}_{i\in\mathcal{R}}, sk_j)$.

(**Anonymity against full key exposure**). For any polynomial-time adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, the ring signature satisfies anonymity if there exists a polynomial related to the integer q_k in λ:

$$\left| \Pr\left[\begin{array}{c} b = b', \\ pk'_{i_0}, \\ pk'_{i_1} \in S \cap \\ \{pk_i^*\}_{i\in\mathcal{R}}. \end{array} \middle| \begin{array}{c} param \leftarrow Setup(\lambda), \\ for\ i \in [1, q_k]: \\ (pk'_i, sk'_i) \leftarrow \\ KeyGen(param, \omega_i), \\ S := \{pk'_i\}_{i=1}^{q_k}, \\ (m^*, \{pk_i^*\}_{i\in\mathcal{R}}, i_0, i_1, St) \\ \leftarrow \mathcal{A}_1^{\mathcal{OS}}(param, S), \\ b \leftarrow_{\$} \{0,1\}, \\ \sigma \leftarrow Sign(m^*, \\ \{pk_i^*\}_{i\in\mathcal{R}}, sk'_{i_b}), \\ b' \leftarrow \mathcal{A}_2(\sigma, \{\omega_i\}_{i=1}^{q_k}, St) \end{array} \right] - \frac{1}{2} \right| \leq negl(\lambda).$$

Threshold ring signature [13]. A TRS scheme is composed of a 4-tuple of polynomial-time algorithms, denoted as **TRS** = (Setup, KeyGen, Sign, Verify). The definitions for Setup, KeyGen remain the same as those used in **RS**. The rest algorithm is defined as follows:

- Sign$(m, \{pk_i\}_{i\in\mathcal{R}}, \{sk_j\}_{j\in\mathcal{S}}) \rightarrow \sigma$. In the signing algorithm for a TRS scheme, a set of t secret keys (indexed by $\mathcal{S}$) is inputted instead of a single secret key. These secret keys correspond to public keys located in the ring, and together they produce a t-out-of-n signature.
- Verify$(m, \{pk_i\}_{i\in\mathcal{R}}, t, \sigma) \rightarrow 0/1$. In the verification algorithm of the TRS scheme, the input consists of the signature, its corresponding message m', the set of ring public keys, and the threshold t. The algorithm is responsible for verifying the legitimacy of the signature.

Recent non-interactive constructions, as described in [14] and [15], include an additional algorithm called CombiSign, which is used to combine partial signatures into a TRS:

$$\text{Combisign}(\text{pp}, \{\sigma_i\}_{i\in\mathcal{S}}, t) \rightarrow \sigma$$

Each signer executes the signing algorithm locally using their own secret key sk_i, resulting in partial signatures σ_i. The algorithm CombiSign takes all the partial signatures $\{\sigma_i\}_{i\in\mathcal{S}}$ and the threshold t as input, and outputs the combined signature.

2.2 Argument of Knowledge

Argument of Knowledge (AoK) [16]. An argument is defined as a tuple of probabilistic polynomial time algorithms (Setup, $\mathcal{P}, \mathcal{V}$), where Setup is the generator of the common reference string, $\mathcal{P}$ is the prover, and $\mathcal{V}$ is the verifier:

- Setup$(1^\lambda) \rightarrow cr$. Output a common reference string cr.
- $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle \rightarrow tr$. On input s and t, $\mathcal{P}$ and $\mathcal{V}$ outputs a transcript through interacting.
- $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b \in \{0, 1\}$. Write $b = 1$ denotes the verifier accepts, and $b = 0$ denotes reject.

Define a formal language:

$$\mathcal{L} = \{x | \exists \omega : (cr, x, \omega) \in \mathcal{R}\}$$

where the variables ω and x represent the witness and the set of statements, respectively, in the given relation $\mathcal{R}$.

The AoK (Setup, $\mathcal{P}, \mathcal{V}$) is defined by satisfying *perfect completeness* and *statistical witness-extended emulation*:

(**Perfect Completeness**). For any non-uniform polynomial time adversaries $\mathcal{A}$, the tuple

(Setup, $\mathcal{P}, \mathcal{V}$) satisfies perfect completeness if

$$\Pr\left[\begin{array}{c|c} (cr,u,\omega)\notin\mathcal{R}\ or & cr\leftarrow \mathtt{Setup}(1^\lambda),\\ \langle\mathcal{P}(cr,u,\omega),\mathcal{V}(cr,u)\rangle=1 & (u,\omega)\leftarrow\mathcal{A}(cr)\end{array}\right]=1$$

(**Statistical Witness-Extended Emulation**). For all deterministic polynomial time prover $\mathcal{P}^*$, the property holds for the 3-tuple of algorithms (Setup, $\mathcal{P}, \mathcal{V}$) if there exists a polynomial time emulator $\mathcal{E}$ such that for any pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$, the following holds:

$$\begin{aligned}&\Pr\left[\mathcal{A}_1(tr)=1 \;\middle|\; \begin{array}{c} cr\leftarrow\mathtt{Setup}(1^\lambda),\\ (u,s)\leftarrow\mathcal{A}_2(cr),\\ tr\leftarrow\langle\mathcal{P}^*(cr,u,s),\mathcal{V}(cr,u)\rangle\end{array}\right]\\ \approx&\Pr\left[\begin{array}{c} \mathcal{A}_1(tr)=1\\ \wedge(tr\text{ is accepting}\\ \Rightarrow(cr,u,\omega)\in\mathcal{R})\end{array} \;\middle|\; \begin{array}{c} cr\leftarrow\mathtt{Setup}(1^\lambda),\\ (u,s)\leftarrow\mathcal{A}_2(cr),\\ (tr,\omega)\leftarrow\mathcal{E}^{\mathcal{O}}(cr,u)\end{array}\right]\end{aligned}$$

where the oracle is defined as $\mathcal{O} = \langle\mathcal{P}^*(cr,u,s),\mathcal{V}(cr,u)\rangle$, which allows for rewinding to a specific point and generating new randomness for the verifier $\mathcal{V}$ from that point onwards.

Inner Product Argument (Bulletproof). Bünz et al. [17] introduce an improved AoK to prove inner-product relations. The inner-product argument is an efficient proof system for relations such as

$$\begin{aligned}\{(\mathbf{g},\mathbf{h}\in\mathbb{G}^n, P\in\mathbb{G}, c\in\mathbb{Z}_p;\mathbf{a},\mathbf{b}\in\mathbb{Z}_p^n):\\ P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}\wedge c=\langle\mathbf{a},\mathbf{b}\rangle\}\end{aligned}\quad(1)$$

where the prover $\mathcal{P}$ tries to convince the verifier $\mathcal{V}$ that c denotes the dot product of vectors a and b, without giving away their values directly.

Overview. To reduce the cost of communication, they modified the relation being proved to

$$\{(\mathbf{g},\mathbf{h}\in\mathbb{G}^n, u, P\in\mathbb{G};\mathbf{a},\mathbf{b}\in\mathbb{Z}_p^n): P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}\cdot u^{\langle\mathbf{a},\mathbf{b}\rangle}\}\quad(2)$$

where u is a fixed group element, corresponding to the unknown discrete logarithm of the group generator.

They achieved the proof system by a recursive alogorithm.

Intuitively, P can be parsed as folllows

$$P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}\cdot u^c=\mathbf{g}_{[:n]}^{\mathbf{a}_{[:n]}}\mathbf{g}_{[n:]}^{\mathbf{a}_{[n:]}}\mathbf{h}_{[:n]}^{\mathbf{b}_{[:n]}}\mathbf{h}_{[n:]}^{\mathbf{b}_{[n:]}}\cdot u^c\in\mathbb{G}\quad(3)$$

In each round, the prover $\mathcal{P}$ computes two commitments $L=\mathbf{g}_{[n:]}^{\mathbf{a}_{[:n]}}\mathbf{h}_{[:n]}^{\mathbf{b}_{[n:]}}\cdot u^{\langle\mathbf{a}_{[:n]},\mathbf{b}_{[n:]}\rangle}$ and $R=\mathbf{g}_{[:n]}^{\mathbf{a}_{[n:]}}\mathbf{h}_{[n:]}^{\mathbf{b}_{[:n]}}\cdot u^{\langle\mathbf{a}_{[n:]},\mathbf{b}_{[:n]}\rangle}$, then sends (L,R) to verifier $\mathcal{V}$ and gets a challenge x. Two proof vectors $\mathbf{a}'$ and $\mathbf{b}'$ of size $n/2$ are computed from x, and set $P'=L^{x^2}PR^{x^{-2}}$ to reconstruct a similar commitment relation of $\mathbf{a}',\mathbf{b}'$ and $\langle\mathbf{a}',\mathbf{b}'\rangle$. Finally, perform the next round algorithm on the given input $(\mathbf{g}',\mathbf{h}',u,P';\mathbf{a}',\mathbf{b}')$ until the size of vectors are recursed to 1. The dot product relation between the original vectors can be verified efficiently using the scalars (a',b') and the group element P' output by the last recursion.

Sum Argument($DualRing_{EC}$). Yuen et al. [12] proposed a DualRing scheme based on Elliptic Curve Cryptography (ECC). In order to optimize the signature size, they introduced a "Sum Argument" algorithm by enhancing the inner product proof algorithm in Bulletproofs. This algorithm is employed to prove the relationship:

$$\left\{(\mathbf{g}\in\mathbb{G}^n, P\in\mathbb{G}, a\in\mathbb{Z}_p;\mathbf{a}\in\mathbb{Z}_p^n): P=\mathbf{g}^{\mathbf{a}}\wedge a=\sum\mathbf{a}\right\}\quad(4)$$

In the proposed schemes in this paper, the Sum Argument algorithm, as introduced by Yuen et al. [12], is also referenced. However, its utilization and purpose in our scheme differ. We represent the sum argument algorithms by the following syntax:

- NISA . Proof$(\mathbf{g},u,P,a,\mathbf{a})\rightarrow\pi$.
- NISA . Verify$(\mathbf{g},u,P,a,\pi)\rightarrow 0/1$.

2.3 Pseudo-random Number Generators(PRNG)

PRNG [18]. A PRNG G is defined as a tuple $(\mathcal{Y},\mu,f,\mathcal{U},g)$, where consists the states set $\mathcal{Y}$, the probability distribution μ on $\mathcal{Y}$ corresponds to the initial seed s_0, the transition function $f: \mathcal{Y}\rightarrow\mathcal{Y}$, the output space $\mathcal{U}$ and the output function $g:\mathcal{Y}\rightarrow\mathcal{U}$. The generator G produces the pseudo-random numbers through the following operations:

1. Input the initial seed $s_0\in\mathcal{Y}$ ground on μ, and the first pseudo-random number is $u_0=g(s_0)$.
2. For each step $i\geq 1$, compute the seed as $s_i=f(s_{i-1})$ and the output value is $u_i=g(s_i)$.

3 Formal Definition

3.1 Syntax

The definition of a non-interactive RTRS scheme consists of a set of 7 PPT algorithms denoted by **RTRS** = (Setup, Keygen, Sign, Verify, Split, SplitCheck, Clip). The first four algorithms are inherited from TRS, whereas the remaining three algorithms permit signatories to revoke their own signatures, authenticate the identity of the revoker, and clip anonymous rings.

- $\text{Split}(\sigma_T, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, sk_{temp}, sk \in \{sk_j\}_{j\in\mathcal{S}}) \rightarrow \sigma$
- $\text{SplitCheck}(m, \{pk_j\}_{j\in\mathcal{S}}, \{pk_i\}_{i\in\mathcal{R}}, (sk, r), \sigma) \rightarrow L_{sp} = \{pk_j\}_{j\in\mathcal{S}'\subset\mathcal{S}}$.
- $\text{Clip}(m, (sk, r), pk_{temp}, \{pk_i\}_{i\in\mathcal{R}}, k, \sigma) \rightarrow \sigma$

where SplitCheck allows any signer of a signature to query the identity of the former signers who have abandoned their signing and return their public keys.

3.2 Security model

EUF-CMA for RTRS [14]. An existentially unforgeable under chosen-message attack (EUF-CMA) RTRS scheme is defined as follows: Let $\mathcal{A}$ be any probabilistic polynomial-time (PPT) adversary. The probability that $\mathcal{A}$ wins the following game against the challenger $\mathcal{C}$ is considered negligible:

- **Setup**. The challenger $\mathcal{C}$ performs $\text{Setup}(1^\lambda) \rightarrow pp$ and $\text{KeyGen}() \rightarrow (sk_i, pk_i)$ for $\mathcal{R} = \{i | i = 1, ..., n\}$ to generate the public parameters along with a set of n key pairs. Then $\mathcal{C}$ transmits pp and $pk_1, ..., pk_n$ to $\mathcal{A}$. Initialize empty sets: L_{corr}, L_{sign}.
- **Queries.** The adversary $\mathcal{A}$ can make polynomially many queries to the following oracles:
  - *OSign.* $\mathcal{A}$ selects a message m and provides it to $\mathcal{C}$. The challenger runs $\text{Sign}(m, \{pk_i\}_{i\in\mathcal{R}}, \{sk_j\}_{j\in\mathcal{S}}) \rightarrow \sigma$ and sends σ to $\mathcal{A}$. Add to L_{sign} : $L_{sign} = L_{sign} \cup (m, \sigma, \mathcal{S})$
  - *OCorrupt.* On input a pk and its index i, this oracle returns the corresponding secret key sk_i. Add to L_{corr} : $L_{corr} = L_{corr} \cup (i)$ The number of corrupted queries should not exceed the signature threshold.
- **Forge**. The game is won by $\mathcal{A}$ if they can produce a valid RTRS for message m^*, where m^* has not been queried.

Signer Ambiguity for RTRS. We say that a RTRS scheme is signer ambiguous if for any PPT adversary $\mathcal{A}$, it is infeasible to disclose which subset of t signers participated in generating σ. The chance of adversary $\mathcal{A}$ winning the game against challenger $\mathcal{C}$ is so small as to be considered negligible:

- **Setup**. Same as above.
- **Queries 1.** Same as above.
- **Challenge**. Adversary $\mathcal{A}$ selects a message m^* and different groups of t signers $PK_0 = (pk_{j_{0,1}}, ..., pk_{j_{0,t}})$, $PK_1 = (pk_{j_{1,1}}, ..., pk_{j_{1,t}})$, both of which are subsets of ring $\{pk_i\}_{i\in\mathcal{R}}$, and have not been queried in Queries 1 phase. $\mathcal{A}$ gives them to $\mathcal{C}$. A bit b is randomly selected from the set $\{0, 1\}$. $\mathcal{C}$ runs signing algorithm to sign m^* and get σ_b, using a set of secret keys corrresponding to one of the signer groups PK_b. Then give σ_b to $\mathcal{A}$.
- **Queries 2.** Same as above, except that PK_0 and PK_1 have not been queried before.
- **Guess.** $\mathcal{A}$ provides a guess b'. The adversary $\mathcal{A}$ is considered to have won the game if the probability that $b' = b$ is greater than $1/2$.

Revoker Anonymity for RTRS.

- **Setup**. Same as above.
- **Queries 1.** Same as above, with additional oracle *ORevoke*.
  - *ORevoke.* The adversary $\mathcal{A}$ transmits a tuple (m, σ) to the challenger $\mathcal{C}$. The challenger then executes the function $\text{Split}(\sigma, m, \cdot, sk \in \{sk_j\}_{j\in\mathcal{S}})$,producing σ^* as the output. Then sends σ^* to $\mathcal{A}$.
- **Challenge**. $\mathcal{A}$ selects a message m^*, a groups of signers $PK = (pk_1, ..., pk_t)$, which is subset of ring $\{pk_i\}_{i\in\mathcal{R}}$, and $pk_{b_0}, pk_{b_1} \in PK$ and gives them to $\mathcal{C}$. The index b is randomly selected from the set $\{b_0, b_1\}$. $\mathcal{C}$ runs signing algorithm to sign m^* by PK to get σ, and revoke pk_b from the σ and output a $(t-1)$-out-of-n signature σ_b^*. Then give σ_b^* to $\mathcal{A}$.
- **Queries 2.** Same as above with *ORevoke*, except that pk_0 and pk_1 have not been revoked before.

- **Guess.** $\mathcal{A}$ provides a guess b'. The adversary $\mathcal{A}$ is considered to have won the game if the probability that $b' = b$ is greater than $1/2$.

Intuitively, the definition of revoker anonymity is that any attacker cannot distinguish the identity of the revoker, even if they can obtain the signatures both before and after the revocation. The concepts of signer ambiguity and revoker anonymity are defined to maintain the anonymity of the signer and the revoker, respectively, before and after the revocation process.

4 Clippable Ring Signatures

The concept of extensibility for RS proposed by Aranha et al. [9] breaks the limitation that the ring of potential signers cannot be changed once its signature has been produced. Their scheme allows for extending an already formed ring to a larger anonymous set without involving another case, shrinking the set by eliminating several insignificant non-signers, which is the role of our proposed concept of clippability. As the anonymity provided by the RS ensures that any non-signer or third party cannot distinguish between the signer and non-signers in the ring, the act of reducing the security can only be carried out by the signer themselves. The same is true in our subsequent work on threshold ring signatures.

It is worth noting that in the absence of other background information or conditions, such clipping of ring signatures is at the cost of certain anonymity, and its standalone application would compromise the anonymity of the signers. The independent clippability feature may find relevance in scenarios where anonymity requirements are not as stringent, providing a balance between storage efficiency and privacy needs. However, for our subsequent work on threshold ring signatures and their application in electronic voting scenarios, despite not being suitable for standalone use, clippability remains meaningful. In Section 6, when designing the revocable threshold ring signature scheme, we rigorously constrained the conditions under which clipping is applicable.

4.1 Syntax

The Clippable Ring Signature (CRS) scheme is a variant of the ring signature scheme that includes an additional algorithm called Clip. This algorithm enables the signer (in the case of TRS, one of the signers) to reduce the size of the group of potential signers associated with a given signature:

$$\text{Clip}(m, sk_j, sk_{temp}, \{pk_i\}_{i\in\mathcal{R}}, pk_\Delta, pk_{temp}, \sigma) \to \sigma'$$

where σ is the signature of message m on ring $\{pk_i\}_{i\in\mathcal{R}}$ signed by signer's secret key sk_j (corresponds to some $j \in \mathcal{R}$), $\mathcal{R}$ denotes the index set of the ring and pk_Δ denotes one of the ring members to be clipped, with $\Delta \in \mathcal{R} \cap \Delta \neq j$. The pk_{temp} is the signature's public key, which is temporary and not bound to the identity of the signer. The algorithm generates a modified signature σ' of the message m on the shrunken ring $\{pk_i\}_{i\in\mathcal{R}-\Delta}$.

Remark. We need to consider the case of multiple clipping by iterative calls to Clip,referred to as atomization. Atomization of the operation involves manipulating one member of the ring at a time and subsequently completing the operation on multiple members through an iterative algorithm. The concept of atomizing the operation is indeed inspired by the work presented in [9]. In [9], this concept is applied to their proposed Extendable Ring Signatures, while in our paper, we apply this idea to our proposed clippability.

4.2 Clippable Ring Signature Scheme

This subsection describes a modified Clippable Ring Signature (CRS) scheme based on the DL-based "Dualring" construction proposed by Yuen et al. [12]. The reasons for choosing dualring are: A Dualring signature consists of a singular response element and a ring of challenge elements, unlike traditional ring signatures. This feature can significantly save the signature size when the challenge scale is smaller than the response scale. Its ring structure is also more "loose", which is helpful for us to adjust the ring size.

Correctness. The correctness of the CRS scheme relies on DualRing and the following formula:

$$R_1 = g^z \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0} \tag{5}$$

Algorithm 2: CRS_{EC} . Sign & Verify & Clip

Sign-Input : $\mathrm{param}, m, sk_j, sk_{temp}, pk_{temp}, \{pk_i\}_{i\in\mathcal{R}}$
Sign-Output: σ
$\sigma_{dr} \leftarrow DualRing_{EC}.Sign(m, \{pk_i\}_{i\in\mathcal{R}}, sk_j)$;
// $\sigma_{dr} = (c_1, ..., c_n, z)$
set $c_0 = H(pk_{temp})$;
return $\sigma = (c_0, c_1, ..., c_n, z)$;

Clip-Input : $\mathrm{param}, m, sk_j, sk_{temp}, \{pk_i\}_{i\in\mathcal{R}}, pk_\Delta, pk_{temp}, \sigma$
Clip-Output: σ'
if $\mathcal{S} \not\subset \mathcal{R} \vee j \in \mathcal{S}$ **then**
  return $\perp$;
if CRS_{EC} . $\mathrm{Verify}(m, \{pk_i\}_{i\in\mathcal{R}}, \sigma) = 0$ **then**
  return $\perp$;
parse $\sigma = (c_0, c_1, ..., c_n, z)$;
random r_2;
// r_1 can be computed from z by signer
$R'_1 = g^{r_1+r_2} \cdot \prod_{i\neq j, i\neq\Delta} pk_i^{c_i}$;
$c' = H(m, \{pk_i\}_{i\in\mathcal{R}-\Delta}, R'_1)$;
$c'_0 = c' - \sum_{i=1, i\neq\Delta}^{n} c_i$;
$z' = r_1 + r_2 - sk_j \cdot c_j - sk_{temp} \cdot c'_0$;
return $\sigma' = (c'_0, \{c_i\}_{i=1, i\neq\Delta}^{n}, z')$;

Verify-Input : $\mathrm{param}, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma$
Verify-Output: 0 or 1
parse $\sigma = (c_0, c_1, ..., c_n, z)$;
if $c_0 = H(pk_{temp})$ **then**
  return $DualRing_{EC}.Verify(m, \{pk_i\}_{i\in\mathcal{R}}, \sigma\backslash c_0)$;
else
  $c = \sum_{i=0}^{l} c_i$;
  $R_1 = g^z \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0}$;
  $c' = H(m, \{pk_i\}_{i\in\mathcal{R}}, R_1)$;
  if c'=c **then**
    return 1;
  else
    return 0;

$$= g^{r_1+r_2-sk_j\cdot c_j - sk_{temp}\cdot c_0} \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0} \tag{6}$$

$$= g^{r_1+r_2} \cdot \prod_{i\neq j} pk_i^{c_i} \tag{7}$$

As Algorithm 2 shows, our clippable scheme includes a DualRing signature when generating a signature, with an additional component c_0 initialized to $H(pk_{temp})$. If no clipping occurs, it is evidently a DualRing signature. When the signer attempts to clip the ring members, the Clip algorithm first modifies R_1 and c, removing the entries of the members to be clipped. However, this causes a mismatch between the challenge c' and the updated hash value. At this point, the role of c'_0 is to bridge the gap between c' and the hash value (the probability of hash collision considered negligible). To prevent adversaries from manipulating c_0 values arbitrarily, the modified c'_0 value is bound to the secret key of the signature, by z', deliberately avoiding adversaries deducing secret information through z and z' information. Note the update of R'_1, which differs from the original R_1 by introducing one value: a new random value r_2. The final signature $\sigma = (c'_0, c_1, ..., c_l, z')$ includes the updated difference c'_0, the challenge ring after clipping, and the updated response z'. Intuitively, this design is a variation of DualRing.

Our verification algorithm is nearly identical to DualRing's verification. When $c_0 = 0$ and the signature is not clipped, the verification algorithm is exactly the same as $DualRing_{EC}.Verify$. However, when $c_0 \neq 0$, c' needs to be calculated according to the updated formula. According to the correctness analysis provided in the paper, verification only succeeds when the given z in the signature is correct and satisfies $g^z = g^{r_1+r_2} pk_j^{-c_j} pk_{temp}^{-c_0}$.

In terms of the space complexity of signatures, the CRS signature only increases the space complexity by adding an additional challenge value compared to the DualRing signature, and its space complexity is also $\mathcal{O}(n)$. We note that Yuen et al. [12] proposed a space optimization method for $DualRing_{EC}$ by using sum proofs based on improved Bulletproofs to prove the relationship $\sum_i c_i = c$ instead of directly providing the challenge ring $c_1, ..., c_n$ in the signature. The space complexity of such proofs is $\mathcal{O}(\log n)$, thereby optimizing the space complexity of DualRing to $\mathcal{O}(\log n)$. In our CRS, since the challenge ring

information is needed in the computation of the clipping operation, it is not possible to use the sum proof to replace the challenge ring. However, if after some dynamic clipping processes, the signature stops updating, then in the last clipping operation, this optimization can be applied to reduce its storage space.

4.3 Security

4.3.1 Security model

The security definition of our CRS is based on DualRing and introduces an additional clipping oracle $\mathcal{OC}lip$.

- $OClip(m, j, \cdot, \{pk_i\}_{i\in\mathcal{R}}, \{pk_j\}_{j\in\mathcal{R}'}, pk_{temp}, \sigma)$: Clipping oracle that outputs the clipped signature $\sigma' \leftarrow (c_0, c_1, \ldots, c_n, z)$. The corrupted clipping set is denoted as CL.

Allow the oracles accessible to the adversary to include $\mathcal{OC}lip$, producing clipped signatures. Modify the conditions $\{pk_j^*\}_{j\in\mathcal{R}} \in S\backslash CL$.

4.3.2 Security proof

Theorem 1. *In the random oracle model, the CRS scheme is clipping unforgeable with respect to insider corruption if DualRing is unforgeable w.r.t. insider corruption*

Proof. Assume $\mathcal{A}$ is an adversary for the unforgeability of the CRS clipping, and construct an adversary $\mathcal{B}$ to break the unforgeability of DualRing.

Setup phase: $\mathcal{B}$ obtains initial parameters and the public key set from the DualRing challenger and forwards them to $\mathcal{A}$.

Simulation phase: $\mathcal{A}$ queries $\mathcal{B}$ as oracle for arbitrary polynomial times. $\mathcal{B}$ responds as follows:

- $\mathcal{OC}$ is identical to DualRing, returns private key.
- $\mathcal{OS}$: $\mathcal{B}$ queries the challenger to obtain a signature, appends a c_0 initialized to $H(pk_{temp})$, and returns it to $\mathcal{A}$.
- $\mathcal{OC}lip$: $\mathcal{B}$ queries $\mathcal{OC}$ to obtain sk_j and sk_{temp}, executes the Clip algorithm to obtain the clipped signature, and returns it to $\mathcal{A}$.
- H: The simulated random oracle.

Challenge phase: Adversary $\mathcal{A}$ returns a forged clipped signature $(m^*, \{pk_i^*\}_{i\in\mathcal{R}^*}, pk_{temp}, \sigma^* = (c_0^*, c_1, \ldots, c_l, z^*))$.

Adversary $\mathcal{B}$, by including pk_{temp} in the set of public keys without any other changes, can regard σ^* as a DualRing signature with a ring size of $l+1$, signed by sk_{temp}, where the used random value is $r = r_1 - sk_j \cdot c_j + r_2$. This is a trivial yet effective forgery. □

Theorem 2. *In the random oracle model, the CRS scheme provides anonymity if DualRing is anonymous.*

Proof. Intuitively, if not clipped, the CRS signature is identical to the DualRing signature, except for an additional value, c_0. Clearly, the theorem holds trivially in this case. Therefore, our focus is on the anonymity of clipped signatures.

The approach for proving the anonymity of clipped signatures in the CRS is as follows:

Suppose $(\mathcal{A}_1, \mathcal{A}_2)$ are adversaries for the anonymity of clipped signatures in the CRS. We aim to construct adversaries $(\mathcal{B}_1, \mathcal{B}_2)$ to break the anonymity of DualRing. *Setup phase*: $\mathcal{B}_1$ obtains initial parameters and the public key set from the DualRing challenger and forwards them to $\mathcal{A}_1$. *Simulation phase*: $\mathcal{A}_1$ queries $\mathcal{B}_1$ as oracles for arbitrary polynomial times. $\mathcal{B}_1$ responds as follows:

- $\mathcal{OS}(m, \{pk_i\}_{i\in\mathcal{R}}, j, \cdot, pk_{temp})$: $\mathcal{B}_1$ queries the challenger to obtain a signature, appends a cinitialized to $H(pk_{temp})$, and returns it to $\mathcal{A}_1$.
- H: The simulated random oracle.

Challenge phase: $\mathcal{A}_1$ provides $\mathcal{B}_1$ with a message m, a set of public keys $\{pk_i\}_{i\in\mathcal{R}}$, a signature public key $pk_{temp} \notin \{pk_i\}_{i\in\mathcal{R}}$, two indices i_0, i_1, and a clipping index $j \notin \{i_0, i_1\}$. Same challenge phase as DualRing anonymity, $\mathcal{B}_2$ obtains a DualRing signature $\sigma = (c_0, c_1, \ldots, c_n, z)$ and all $\{\omega_i\}_{i=1}^{q_k}$. After receiving the response, $\mathcal{B}_2$ removes the challenge c_j and the public key pk_j from the signature, renumbering the remaining ring members from 1 to $n-1$. $\mathcal{B}_2$ then randomly samples z' and c_0, calculates $R_1 = g^{z'} \prod_{i=1} pk_{temp}^{c_0}$, and sets $H(m, \{pk_i\}_{i\in\mathcal{R}}, R_1) = \sum_{i=0}^{n} c_i$. If the hash value has already been queried from the H oracle, $\mathcal{B}_2$ aborts. $\mathcal{B}_2$ returns $\sigma^* = (c_0, c_1, \ldots, c_{n-1}, z')$ and $\{\omega_i\}_{i=1}^{q_k}$ to $\mathcal{A}_2$.

Output phase: $\mathcal{A}_2$ outputs a guessed bit b'. As the bit b was not utilized in the generation process of the clipped signature σ^*, $\mathcal{A}_2$ can only succeed with a probability of 1/2.

Analysis: According to [12], the probability of simulation success (i.e., the event not aborting) cannot be ignored, and the adversaries $(\mathcal{B}_1, \mathcal{B}_2)$ have no advantage in guessing. As the adversaries $(\mathcal{B}_1, \mathcal{B}_2)$ simulate the challenger for $(\mathcal{A}_1, \mathcal{A}_2)$, for q_h queries to the H oracle and q_s queries to the $\mathcal{SO}$ oracle, the probability of success on the first query is at least $(1 - \frac{q_h}{|\Delta_c|})^2$. This pattern continues, and after q_s queries to $\mathcal{SO}$, the probability of success is at least the square of the success probability in the DualRing simulation.

In our scheme, we assume the anonymity of DualRing, meaning the probability of success in DualRing simulation cannot be ignored. Therefore, if DualRing is anonymous, it is impossible for any PPT adversary to win with more than half of a non-negligible probability. □

Hence, our CRS construct is secure; it satisfies unforgeability and anonymity. Although clipping reduces the anonymity of the RS, it still can hide the signer in a set of potentially anonymous groups.

5 Splittable Threshold Ring Signatures

In non-interactive TRS systems, the Sign & Combine (CombiSign) pattern is often used to assemble threshold signatures. Specifically, Combine extends signers on the same ring to their union. The opposite is a case where a signer in the TRS can choose to demote himself to a non-signer unless there is only one signer left in the ring. Our definition of separability aims to achieve this, reducing the set of signers while ensuring anonymity.

In the e-voting scenario, splittability gives participants a chance to regret. The original signature can be regarded as a pre-vote. After the signature is generated and before the final confirmation by the third party, participants can withdraw their signer identity through Split. Despite this, the signer still belongs to the non-signers in the ring after withdrawal and has the same possibility as any other signer from the verifier's view. The clippability of TRS provides the opportunity to leave and is proven to pose no additional privacy risk to the parties, which will be elaborated on in Section 6.

Based on our improved RS construction, first consider how to convert it into a TRS scheme. Yuen et al. [12] have introduced inner product proof and improved it in their original EC-based Dualring scheme, but the role of sum argument in the signature scheme in our work is quite different. Yuen et al. [12] adopted a unit vector to transform the inner product argument system into the argument for sum relation and used it to compact the size of generated DL-based ring signature from $\mathcal{O}(n)$ into a logarithm. In our construction, we utilize it to ensure that the number of signers is fixed and that the responses are correct in the TRS scheme.

5.1 Threshold ring signature

5.1.1 Transform RS to TRS

When considering the signature of t-out-of-n relations, the similarities and differences with 1-out-of-n relations are that, on the one hand, they still need to prove that the public key of each signer is present in the anonymous ring; on the other hand, to prevent a single or partial of signers from randomly forging joint signatures with other signatories. Simply adding secret keys to the original RS construct is insufficient to prove the absolute number of hiding signers in the ring to the verifier. Hence we need the help of the Sum Argument system to confirm this relationship. Through observation, we found that the computing of z could be seen as a simple sum relationship consisting of $t + 2$ scalars. As shown in Algorithm 3, when provided with the message m, the set of signers' secret keys, signing key pair, and the ring of public keys, the signing algorithm generates a signature of size $\mathcal{O}(n)$. For verification, on inputting the same parameters except for the secret info, the verifying algorithm makes all necessary checking and informs "yes" or "no".

As illustrated in Algorithm 3, the fundamental structure of our TRS remains based on DualRing, with several distinctions:

- Multiple signers are employed rather than a single one.
- The generation of the challenge ring is entirely random, and the difference between the hash value and the sum of the challenges is filled with c_0. Correspondingly, information related to

Algorithm 3: TRS_{EC} . Sign & Verify

Sign-Input : $\mathrm{param}, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \{sk_j\}_{j\in\mathcal{S}}, sk_{temp}$
Sign-Output: σ_T
random r, c_i for all $i \in \mathcal{R}$;
$R = g^r \cdot \prod_{i\in\mathcal{R}-\mathcal{S}} pk_i^{c_i}$;
$c = H(m, \{pk_i\}_{i\in\mathcal{R}}, R)$;
$c_0 = c - \sum_{i\in\mathcal{R}} c_i$;
$z = r - \sum_{j\in\mathcal{S}} sk_j \cdot c_j - sk_{temp} \cdot c_0$;
set
$\mathbf{a} = (r, -sk_{temp}\cdot c_0, -sk_1 \cdot c_1, ..., -sk_t \cdot c_t)$;
$P = \mathbf{g}^{\mathbf{a}} = g^z$;
$\pi \leftarrow \mathrm{NISA}.\mathrm{Proof}(\{\mathbf{g}, u, P, z\}, \mathbf{a})$;
return $\sigma_T = (c_0, c_1, ..., c_n, z, \pi, t = |\mathcal{S}|)$;
Verify-Input : $\mathrm{param}, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma_T$
Verify-Output: 0 or 1
parse $\sigma_T = (c_0, c_1, ..., c_n, z, \pi, t)$;
$R = g^z \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0}$;
$c = \sum_{i=0}^{n} c_i$;
$P = g^z$;
if $\mathrm{NISA}.\mathrm{Verify}(\mathbf{g}, u, P, z, \pi) = 0$ **then**
  return 0;
if $c \neq H(m, \{pk_i\}_{i\in\mathcal{R}}, R)$ **then**
  return 0;
return 1;

the signing key and c_0 is incorporated into the response z. This technique was employed in the preceding section.

- The NISA is used to prove the quantity of secrets utilized during the signing process, specifically the number of signers.

Some readers may notice that the generation phase of signatures in our TRS scheme involves multiple signers collaborating. This process may entail the use of secret sharing to generate the signatures collectively. However, the specific implementation of this secret sharing protocol is beyond the scope of this paper and may serve as a potential research direction for future work.

5.2 Syntax

A STRS scheme is a type of TRS scheme with an additional algorithm, Split, which allows one of the signers to revoke his own authentication and degrade to a non-signer ring member. When provided with a t-out-of-n signature σ_T of message m on ring $pk_i i \in \mathcal{R}$, as well as the secret key sk of one of the signers, the algorithm produces a $(t-1)$-out-of-n signature σ^* where sk has been revoked and degraded.

$$\mathrm{Split}(\sigma, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, sk_{temp}, sk \in \{sk_j\}_{j\in\mathcal{S}}, r) \rightarrow (\sigma_T, \{\sigma^*\})$$

where $\mathcal{S}$ denotes the the set of indices corresponding to the signers' public keys, and $\mathcal{S} \subset \mathcal{R}$.

5.3 Splittable Threshold Ring Signature Scheme

For TRS constructions using Sign &Combine (CombiSign) pattern, CombiSign is usually a simple connection of the one-signer signatures generated separately on the same ring. In this case, Split naturally exists, as simple as its generation process: cutting the signature into two segments. If the TRS scheme is constructed this way, then the RTRS scheme can be constructed directly. We focus on other cases; in some interactive TRS, it is a challenge to design a secure splitting scheme. For example, the splitting algorithm of our proposed TRS scheme can be constructed as shown in Algorithm 4.

When a signer wants to revoke, all the auxiliary information he can give including the randomness of signature and his secret key, and his knowledge of who the real signers are. The algorithm proceeds as follows. Firstly, essential validations are conducted to ensure the input signature is legitimate. Subsequently, the revoker can utilize their private key to generate a new response. It takes a form similar to the response in a ring signature, but it is used to nullify the authorization concerning the revoker in the original TRS. Correspondingly, based on the variations in the random numbers used and the signers' set, the new values for R_1^* and c_0^* are sequentially computed. The Sum Argument also requires updating to prove that the new TRS has $t-1$ signers. The computation of the revoker's linkable tag is performed to identify double-splitting. Finally, the newly generated values in the above process are appended to the original signature as σ^*. If the Split operation is

Algorithm 4: STRS_{EC} . Split

Input: $\sigma, m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, sk_{temp}, sk_\Delta \in \{sk_j\}_{j\in\mathcal{S}}, r$
Output: σ

// generate $1-out-of-(n-t+1)$ RS
parse $\sigma_T = (c_0, c_1, ..., c_n, z, \pi, t)$;
if $\mathrm{STRS}_{EC}.\mathrm{Verify}(m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma) = 0$ **then**
  return $\perp$;
random r^*;
$R_1^* = g^{r-r^*} \cdot \prod_{i\in\mathcal{R}-\mathcal{S}+\Delta} pk_i^{c_i}$;
$c_0^* = H(m, \{pk_i\}_{i\in\mathcal{R}}, R_1^*) - \sum_{i\in\mathcal{R}} c_i$;
$z^* = r^* - sk_\Delta \cdot c_\Delta - sk_{temp} \cdot c_0^*$;
$\mathbf{a}^* = (r^*, -sk_\Delta \cdot c_\Delta, -sk_{temp}, c_0^*)$;
$P^* = \mathbf{g}^{\mathbf{a}^*} = g^{z^*}$;
$\pi^* \leftarrow \mathrm{NISA}_{IPr}.\mathrm{Proof}(\{\mathbf{g}, u, P^*, z^*\}, \mathbf{a}^*)$;
$I = H_{\mathbb{G}}(pk_\Delta)^{sk_\Delta}$;
add $\sigma^* = (c_0^*, z^*, \pi^*, I)$ to $\{\sigma^*\}$;
return $\sigma = (\sigma_T, \{\sigma^*\})$.

Algorithm 5: STRS_{EC} . Verify

Input: $m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, (\sigma_T, \{\sigma^*\})$
Output: 0 or 1

parse $\sigma = (\sigma_T, \{\sigma^*\})$;
if $\mathrm{TRS}_{EC}.\mathrm{Verify}(m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma_T) = 0$ **then**
  return 0;
for *all* $\{\sigma^*\}$ *in set* **do**
  // check signatures for link
  if $\exists I_i = I_j, i \neq j$ **then**
    return 0;
  // Verify*: compute R_1 with $R_1 = g^{z-\sum z^*} \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0-\sum c_0^*}$ instead.
  if $\mathrm{TRS}_{EC}.\mathrm{Verify}^*(m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma^* \cup \{c_1, ..., c_n, z\} \cup \{(z^*, c_0^*)\}) = 0$ **then**
    return 0;
return 1.

executed multiple times, each updated signature is appended to the original signature, forming the set $\{\sigma^*\}$. By adding to the initial threshold signature, any verifier can be persuaded that some member of the signers has joined the non-signer collection anonymously. To achieve those hiding of the new ring, we modified the original verification algorithm of ring signature to Verify*, where R_1 is computed by $g^{z-z^*} \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i}$ instead.

Algorithm 5 demonstrates the verification procedure of STRS, which includes regular TRS verification along with the additional ring signature verification for splitting (if it exists). The verifier screens out linked signatures by comparing the "key image" I.

The correctness can be checked as

$$R_1 = g^{z-z^*} \cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0-c_0^*} \tag{8}$$

$$= \left(g^{r-r^*} \cdot pk_\Delta^{c_\Delta} \cdot pk_{temp}^{c_0^*-c_0} \cdot \prod_{i\in\mathcal{S}} pk_i^{-c_i} \right) \tag{9}$$
$$\cdot \prod_{i\in\mathcal{R}} pk_i^{c_i} \cdot pk_{temp}^{c_0-c_0^*}$$

$$= g^{r-r^*} \cdot \prod_{i\in\mathcal{R}-\mathcal{S}+\Delta} pk_i^{c_i} \tag{10}$$

Let's consider whether malicious ring members (including non-signers and signers attempting to forge the actions of other signers) can claim that a signer has performed a revocation by forging an additional ring signature. Attackers fall into several possible categories, each with the information they possess: firstly, for malicious non-signers, they only have their own private keys and some public information from the original ring signature. More potent attackers, the malicious signers, attempt to forge revocations of other co-signing members. They also have additional secrets, including the real signer identity set and the random secrets used in the original signature. Finally, conspiracies involving both malicious signers and non-signers must be considered.

When attackers attempt to forge a ring signature using the Split algorithm, they encounter the following issues: Attackers can generate a seemingly normal ring signature by customizing values for r, r^*, sk_Δ, including a normally verifiable Sum

Argument π^*. However, only when using the correct r, signer private key sk_Δ, and signers set, can R_1 be calculated to recover (as implied by signature correctness), and the ring signature be successfully verified. Clearly, any malicious non-signer, signer, or their collusion cannot collectively possess this knowledge, resulting in any forgery leading to verification failure.

In conclusion, based on our design:

1. Signers cannot revoke signatures from others in the ring, although no verifiers can discern the revoker's identity from the output signature.
2. Non-signers in the ring cannot generate a valid revocation to pass the verification of the output signature.

Linkability. What is the method for confirming that two output signatures were not produced by the same signer? For instance, when there are multiple π^* in the set, it signifies that multiple signers have requested to revoke their authentication. However, when a single signer generates multiple ring signatures using distinct randomness due to malicious intent or communication issues, it can potentially mislead the verifier into believing multiple signers have been revoked. To mitigate this issue, introducing linkability in the ring signature scheme can be an effective solution. With linkability, the verifier can easily detect signatures from the same signer by comparing relevant parameters.

5.4 Security Proof

Since the proof process is similar to Theorem 1 and Theorem 2, we give the sketch of the proofs to save space.

Theorem 3. *In the random oracle model, the STRS scheme is EUF-CMA.*

Proof. Suppose an adversary $\mathcal{A}$ can break the unforgeability of STRS (Splittable Ring Signature) scheme. In that case, we can construct an algorithm, denoted as $\mathcal{B}$, that enables the breaking of the unforgeability of DualRing-EC.

If adversary $\mathcal{A}$ makes a signing oracle query, $\mathcal{B}$ responds by making a DualRing signing oracle query, from which it obtains a signature $\sigma = (c_1, ..., c_n, z)$ of the signer j. $\mathcal{B}$ get signers's secret key set through *OCorrupt*, randomly samples r', and updates r to $r' + r + \sum_{i \in \mathcal{S} \setminus j}$, then compute $R' = g^{r'} \cdot R$, $c' = H(m, \{pk_i\}_{i \in \mathcal{R} \cup \{temp\}}, R'), c_0 = c' - c, z' = z + r' - sk_{temp} \cdot c_0, P = g^z$, and the proof π in turn. $\mathcal{B}$ then returns to $\mathcal{A}$ a TRS signature $\sigma' = (c_0, c_1, ..., c_n, z', \pi, t)$ to simulate the output of the signing oracle.

During the challenge phase, if $\mathcal{A}$ produces a forgery, $\mathcal{B}$ is capable of extracting the secret vector using the statistical witness-extended emulation property of NISA. This forged signature can also be regarded as a DualRing ring signature, constituting a valid forgery against $DualRing_{EC}$.

Furthermore, when revocation is involved in the signature, our Split algorithm is designed to retain the original TRS signature while appending a new TRS. The appended signature σ^* can also be verified as a ring signature. In conclusion, if adversary $\mathcal{A}$ can successfully forge STRS, then an adversary $\mathcal{B}$ can be constructed to break the unforgeability of DualRing signatures. □

Theorem 4. *In the random oracle model, the STRS scheme is signer-ambiguous.*

Proof. Suppose $\mathcal{A}$ is an adversary capable of breaking the signer ambiguity of the STRS protocol. Then, an algorithm $\mathcal{B}$ can be constructed to break the anonymity of DualRing. When $\mathcal{A}$ queries the signing oracle, $\mathcal{B}$ generates a simulated signature σ^* in a similar way as in the proof of unforgeability and sends it to $\mathcal{A}$.

During the challenge phase, $\mathcal{A}$ chooses a message m^*,the public key set $\{pk\}_{i \in \mathcal{R}}, pk_{temp}$, and the sets of signers' public keys $PK_0, PK_1 \subset \{pk\}_{i \in \mathcal{R}}$ and sends them to $\mathcal{B}$, $t = |PK_0| = |PK_1|$. Same challenge phase as DualRing anonymity, $\mathcal{B}$ computes a DualRing signature $\sigma = (c_0, c_1, ..c_n, z)$. $\mathcal{B}$ randomly samples a z' and calculates R' using the formula in $TRS_{EC}.Verify$, setting $H(m, \{pk\}_{i \in \mathcal{R}}, R') = \sum_{i=0}^{n} c_i^*$. The process terminates when a collision occurs with a value previously queried to the hash oracle H. z' is randomly split into $t+2$ parts, and a proof π is generated by executing $NISA.Proof$. The signature $\sigma^* = (c_0, ..., c_n, z', \pi, t)$ is then returned to $\mathcal{A}$. $\mathcal{A}$ subsequently returns a guess bit b' to $\mathcal{B}$. Since $\mathcal{B}$ did not use b in the signature generation process, the probability of $\mathcal{A}$ guessing correctly can only be 1/2.

Regarding the probability of termination due to hash collisions, please refer to the proof section of Theorem 2. When the signature includes a revocation mechanism, the obtained signature is an

attachment of a linkable TRS on the same ring in TRS. The simulation principle for its challenge part is similar. In this case, the split operations do not affect the signer ambiguity of STRS. □

Theorem 5. *In the random oracle model, the STRS scheme achieves revoker anonymity.*

Proof. Based on the signer-ambiguous property of STRS, an attacker is unable to distinguish signatures generated by different sets of signers. As a specific instance, the additional signature σ^* in STRS is a linkable TRS attached to the original signature on the same ring, with the only distinction being the presence or absence of a revoker in the signer set. Trivially deduced from the fact that STRS satisfies signer-ambiguous, an attacker is further unable to differentiate the changes in the signer set caused by the Split operation to identify the revoker. □

Hence, our STRS construct is secure; it satisfies unforgeability, revoker anonymity, and signer ambiguity.

6 Revocable Threshold Ring Signature Scheme

The TRS is a cryptographic building block that allows multiple signers to collaboratively generate a signature while maintaining anonymity by hiding the signers' identities within a ring of potential signers. Existing schemes are limited in that they do not allow for resizing the ring after signature generation. However, Aranha et al. [9] proposed a method that overcomes this limitation by achieving the signers' flexibility and the ring size's extendability. Compared to extendability, revocability naturally does not require consideration of external members to the ring. However, additional issues must be taken into account to reduce the ring's size and the number of signers while maintaining the anonymity of the primary signers. Although we have discussed and presented solutions for these properties separately in the previous sections, our proposed revocability concept aims to integrate these concerns and preserve the anonymity of the original signers.

6.1 Boundaries for Anonymity

As we all know, for a t-out-of-n RS, the signature is co-signed by t signers and hides them in an anonymous set of size n. Each signer's anonymity level can be computed: t/n. The objective of our study is to ensure that any signer's anonymity is maintained at not lower than this level throughout the entire lifecycle of the signature during all operations.

Remarks on clipping. To execute `Clip`, the following items must be observed

1. Limit anonymity level. The precondition for clipping has not been discussed in the context of CRS. When it comes to TRS, it is clear that clipping non-signers in the ring can compromise the anonymity of the signature. However, to maintain anonymity, never lower than the original signature, clipping can only be made by other signers after some signers revoke(by Split). Let $k = n - n^*$ be the number of ring members to be clipped, and let α be the number of split signers, where $\alpha = 1, 1 \leq t \leq n, 1 \leq k \leq n-t+1$. In this case, the inequality $(t-1)/n < (t-1)/(n-k) \leq t/n$ holds. As shown in Figure 3, the anonymity of the original (t, n) signature is 3/8. When signer s_6 revokes its signature, the anonymity increases to 2/8. If other signers clip no more than one non-signer in the ring, the anonymity drops to 2/7, which is still higher than the original value. Therefore, the number of nodes that can be clipped in a TRS signature depends on its original probability, the ring size, and the signers amount who revoked their signature before clipping.
2. Prevent redundant call attacks. We emphasize that the clipping of ring members can only be carried out in a pseudo-randomly, based on some non-sequential method. Unlike in RS, where clipping can only be performed by the sole signer, any signer in TRS can clip their common signature. If signers clip non-signers in different or entirely random ways, when redundant clipping is executed, the number of declared non-signers may exceed what is expected, leading to an excessive decrease in anonymity. Figures 3 (c) and (d) illustrate that signers s_7 and s_8, respectively, suggest removing non-signers s_3 and s_6 from the ring. Each clipping generates a signature with an

anonymity level of at least 3/8 of the original signature requirement. However, if both clippings occur simultaneously, it would result in the exposure of the identities of both non-signers, reducing the actual anonymity level observed by the verifier to 2/6, which is lower than the original signature requirement. Moreover, clipping in a completely sequential manner is also not feasible, as by analyzing the skipped members in the ring, an attacker can quickly figure out the actual signer's index.

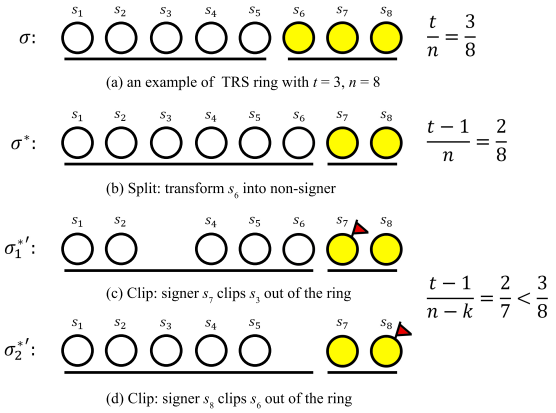


Fig. 3: Examples of signer `Split` and `Clip`

Clipping boundaries

1. $\alpha = 1, k = 1$. Only clip one member. The anonymity of the split signer hiding among the non-signers, or the probability that the clipped member is a former signer, is $\frac{1}{n-t+1}$. Although no longer a signer, the anonymity still needs to be protected, so it is stipulated $\frac{1}{n-t+1} \leq \frac{t}{n}$, which is always true when $1 \leq t \leq n$. The equation holds when $t = 1$ or n; however, in practice, this scenario is unlikely to occur.
2. $\alpha > 1, k = 1$. Similarly, $\frac{\alpha}{n-t+\alpha} \leq \frac{t}{n}$ holds true when $\alpha \leq t \leq n$ (naturally true in TRS).
3. $\alpha = 1, k > 1$. The core idea is still to guarantee $k \cdot \frac{1}{n-t+1} \leq \frac{t}{n}$. When $t = \frac{n+1}{2}$, the rearranged $t^2 - (n+1)t + kn \leq 0$ is most likely true. Substitute t and the inequality can only be true if $n^2 + (2-4k)n + 1 \geq 0$. So we ask $k \leq \frac{(n+1)^2}{4n}$, the maximum value of k is the largest positive integer within this range.
4. $\alpha > 1, k > 1$. Similarly, set $t = \frac{n+\alpha}{2}$ and the maximum possible value of k is the greatest positive integer that satisfies the inequality $k \leq \frac{(n+\alpha)^2}{4n\alpha}$.

Remarks on anonymity. The definition of anonymity in [10] includes two aspects: Stronger Anonymity, in contrast to [9], attackers can observe all intermediate signatures, not just the final ETRS. Fellow-signer Anonymity, also known as inter-signer anonymity, where co-signers remain anonymous to each other.

As mentioned in Section 1.1, our proposed revocable signature is "inherently" immune to the weaknesses in [9] because:

1. In our scheme, not only is the anonymity of signers protected, but the revoked signer's anonymity is always maintained. Specifically, the Split algorithm declares the revocation of a signer in the ring without revealing the revoker's identity. The Clip algorithm, while declaring the departure of a non-signer from the ring, cannot ascertain the identity of this member in the initial signature. Thus, the evolution of the signature does not compromise anonymity. Quantitative calculations of the impact on anonymity have been provided above.
2. From the perspective of algorithm construction, our Split and Clip algorithms achieve their objectives by appending information to the original signature rather than altering the original signature. Therefore, even if adversaries have access to the "full evolution" of signatures, they cannot obtain more information than the last signature. Hence, we describe our design's immunity to this weakness as "inherent".

So, in terms of the anonymity level of our scheme, it intuitively meets the stronger anonymity requirements outlined in [10], surpassing the anonymity in [9] but falling short of the strongest anonymity in [10]. The reasons for the inapplicability of inter-signer anonymity in the design of the revocable scheme will be explained in Section 6.2.

6.2 Construct DL-based RTRS

All members in a ring signature are indistinguishable for any verifier, but SplitCheck requires signers in the ring can learn which signers have

revoked their authorization. Additionally, Clip attempts to provide signers with the ability to delete members from the ring without interaction, thereby reducing the anonymity of the signature. To achieve these goals, all signers must have knowledge of some additional advantages, which are the original signers set and the randomness used to sign.

Our clippability rules out the possibility of anonymity among signers, as signers can use the Clip algorithm as an oracle to query the identities of co-signers through the SplitCheck algorithm. In contrast, in the corresponding extensibility scheme [9], anonymity among signers is maintained. You might question the rationale behind our design and utilization of SplitCheck in RTRS. Firstly, we de-anonymize among signers to facilitate non-interactive revocation operations for individual signers. Regarding extendable signatures [9], they facilitate the expansion of the ring size without compromising the anonymity of the signers when additional non-signers are added to the ring; instead, it has the potential to enhance the overall anonymity of the signature. However, the situation becomes considerably more complex when considering a reduction in the ring size. Allowing anonymous signers to adaptively decrease anonymity is evidently not feasible. Secondly,while our design sacrifices anonymity among signers, it comes with practical benefits. Consider a scenario where actual signers engage in the signing process without knowledge of their co-signers, and the resulting signature is subsequently independently modified and reissued multiple times by anonymous participants. For signers, a signature that has been altered by anonymous participants may raise trust issues.

To implement SplitCheck, the participant who performs Split must leave information that other signers can use to verify their identity while ensuring that no knowledge would be leaked to anyone else. This can be achieved through simple knowledge proof. To achieve this, a new parameter s is calculated in algorithm RTRS . Split as following:

$$s = r + sk_{\Delta} \cdot H(pk_{\Delta} \| r) \tag{11}$$

Then add it to the generated signature σ^*:

$$\sigma^* = (c_0^*, z^*, \pi^*, I, s) \tag{12}$$

Algorithm 6: RTRS_{EC} . SplitCheck

Input: $m, \{pk_j\}_{j\in\mathcal{S}}, \{pk_i\}_{i\in\mathcal{R}}, (sk, r), \sigma$
Output: L_{sp}

```
   // all signers know the original
      signing members and the
      randomness r
 1 if g^sk ∉ {pk_j}_{j∈S} then
 2     return ⊥;
 3 parse σ_T = (σ, Σ_s = {σ_l^*});
 4 if Σ_s = ∅ then
 5     return ∅;
 6 parse σ_T = (c_0, c_1, ..., c_n, z, π), σ_l^* =
     (c_0^*, z^*, π^*, I, s_l);
 7 if
     RTRS_EC . Verify(m, {pk_i}_{i∈R}, pk_temp, σ) =
     0 then
 8     return ⊥;
 9 set L_sp = ∅;
10 for l do
       // l is the index of signatures
          in set {σ^*}
11     for j ∈ S do
12         if pk_j = g^sk then
13             return ⊥;
14         else if g^{s_l − r} = pk_j^{H(pk_j ‖ r)} then
15             add pk_j to L_sp;
16 if |L_sp| ≠ |Σ_s| then
17     return ⊥;
18 return L_sp.
```

As Algorithm 6 shows, after a series of validity checks, the signers who have passed can search for the revoked signers as in line 12 and return a set of splitter public keys. Since all signers share the initial random value r of the signature, any third party or non-signer member of the ring cannot impersonate the signer and obtain valid splitting information; even for the former signers who have left the ring due to clipping, they cannot pass the legitimacy check.

Algorithm 7 presents our clip algorithm for RTRS. In contrast to the clip in the ring signature we proposed above, the clip in RTRS

Algorithm 7: RTRS_{EC} . Clip

Input: $m, (sk, r), pk_{temp}, \{pk_i\}_{i\in\mathcal{R}}, k,$
$\sigma = (\sigma_T, \{\sigma^*\})$
Output: σ

if RTRS_{EC} . $\mathrm{Verify}(m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, \sigma) = 0$ **then**
  return $\perp$;
RTRS_{EC} . $\mathrm{SplitCheck}(m, \{pk_j\}_{j\in\mathcal{S}}, \{pk_i\}_{i\in\mathcal{R}}, (sk, r), \sigma) \to L_{sp}$;
$\alpha = |L_{sp}|$ **if** $\alpha = 0$ **then**
  return $\perp$;
index set $\bar{\mathcal{R}} = (\mathcal{R}/\mathcal{S}) \cup L_{sp}.index$;
clipped index set $L_c = \{j\}$;
// Empty set when no clipping occurs
if $K + k > \frac{t(n-t+\alpha)}{n\alpha}$ **then**
  return $\perp$;
$u = |\bar{\mathcal{R}}|$;
for k **do**
  $p = (\mathrm{PRNG}.G(r))\%u$;
  add j to L_c where $j \in \bar{\mathcal{R}}$ is the pth index;
  $R' = g^{z-z'-\sum z^*} \prod_{i\in\mathcal{R}-L_c} pk_i^{c_i} \cdot pk_{temp}^{c_0-c_0'-\sum c_0^*}$;
  // compute new (c_0', z') for the participant being clipped.
  $c_0' = c_0' + H(m, \{pk_i\}_{i\in\mathcal{R}-L_c}, R') - \sum_{i\in\mathcal{R}-L_c} c_i$;
  $z' = z' + r' - sk_{temp} \cdot c_0'$;
  $K = K + 1$;
  update $\sigma' = ((c_0', z'), L_c, K)\}$;
  $u = u/j$;
$\sigma_T' = \sigma_T \backslash \{c_j\}_{j\in L_c}$;
return $\sigma = (\sigma_T', L_\sigma = (\{\sigma^*\}, \sigma'))$.

needs to restrict the clipping authority of the signer to protect anonymity. The executor of our clipping algorithm is an actual signer. However, since revocation and clipping operations are both non-interactive, this executor first needs to grasp the current state of the signature to determine the clipping order, including information such as which original signers revoked their signatures, the nodes that have already been clipped, and their quantities. Moreover, when a revoked signer attempts to forge a clipping, it cannot pass verification. Our Clip algorithm achieves this by invoking the SplitCheck algorithm.

Specifically, in addition to legitimacy verification, the content of Algorithm 7 comprises the following three major parts:

- Invoking SplitCheck to obtain information about the revokers.
- Verifying that the clipping aligns with the anonymity level requirements.
- Determining the clipping order, calculating the values $\{(c_0', z')\}$ required for updating the signature, and updating the signature.

The correctness of Algorithm 7 relies on:

$$R' = g^{z-z'-\sum z^*} \prod_{i\in\mathcal{R}-L_c} pk_i^{c_i} \cdot pk_{temp}^{c_0-c_0'-\sum c_0^*} \quad (13)$$

$$= g^{r-\sum r^*-\sum r'} \prod_{i\in\mathcal{R}-\mathcal{S}+\{\Delta\}-L_c} pk_i^{c_i} \quad (14)$$

Correspondingly, the verification algorithm must be adjusted by incorporating some legitimacy checks, as shown in Algorithm 8. The primary idea of the verification algorithm is to validate the TRS formed by the updates in signatures σ^* and σ'. Before this, a series of legitimacy checks are conducted, including verification of all NISA proofs and validation of linkability. Failure in any of these checks results in signature rejection. Additionally, legitimacy verification of clipping scale is performed, as excessively clipped signatures are inherently illegitimate.

Regarding the clipping sequence, we require a fixed clipping series that satisfies specific properties: it should be unpredictable but reproducible, cover all nodes, avoid clustering, and prevent additional computations caused by conflicts. We were inspired by the pseudo-random sequence method for conflict resolution in search algorithms, and we used a PRNG to sample values in the set of non-signer nodes, using the secret random value shared by all signers as the initial seed for the PRNG to obtain a clipping sequence that signers can reproduce.

One possible question is how to deal with the potential mutual influence between Clip and Split. For Split, even if the signature has been split before, other signers only need to add their

Algorithm 8: RTRS_{EC} . Verify

Input: $m, \{pk_i\}_{i\in\mathcal{R}}, (\sigma_T, \{\sigma^*\})$
Output: 0 or 1

```
// Sequentially verify the
   signatures in L_σ, ensuring the
   legality of each operation step.
```
for *every signature in* L_σ **do**
  $int\ \alpha = 0$;
  for *all* $\{\sigma^*\}$ *in set* **do**
    `// Verify the proofs.`
    if $\mathrm{NISA}.\mathrm{Verify}(\mathbf{g}, u, P, z, \pi) = 0$ **then**
      return 0.
    `// check signatures for link`
    if $\exists I_i = I_j, i \neq j$ **then**
      return 0;
    else
      $\alpha++$;
  if $K > \frac{t(n-t+\alpha)}{n\alpha}$ **then**
    return 0.
  `// Verify the lastest signature which is` σ^* `or` σ' `in set`
  `// Verification for proof is not required.`
  if $\mathrm{TRS}_{EC}.\mathrm{Verify}(m, \{pk_i\}_{i\in\mathcal{R}}, pk_{temp}, (c_0 - c_0' - \sum c_0^*, \{c_i\}_{i\notin L_c}, z - z' - \sum z^*)) = 0$ **then**
    return 0.
return 1.

signed σ^* to the signature when proposing split without modifying the existing parts of the signature. Regarding Clip, signers achieve clipping by updating the common values for the signature. If Clip and Split are executed alternately, it is only necessary to update the corresponding ring and include (z', c_0') and all (z^*, c_0^*) when computing R. The effectiveness of this approach lies in the fact that our proposed Clip and Split algorithms do not disrupt the verification relationship of DualRing, i.e., $R = g^r \prod_{i\in\mathcal{R}-\mathcal{S}} pk_i^{c_i} = g^z \prod_{i\in\mathcal{R}} pk_i^{c_i}$. We only make adjustments to the common parameters (c_0, z) and the ring members, while still maintaining the validity of the relationship.

Considering the impact of clipping different nodes on probability, although the clipped nodes are either degraded from the previous signer or initially as the non-signer, there are differences in the anonymity probability of the following clipping. However, the verifier cannot distinguish the identity of the declared nodes. Therefore, in practice, the impact of clipping different non-signing nodes in the current signature on the anonymity of the signature is indistinguishable.

7 Implementation

We implement the proposed CRS and TRS constructs, as well as the ERS and DualRing schemes using NIST-192. The hardware support platform is 12th Gen Intel(R) Core(TM) i5-12500H 2.50GHz. Each algorithm in the experiments was executed 10^3 times and averaged to obtain stable results. The average time costs of signature generation are shown in Figure 4 and 5.

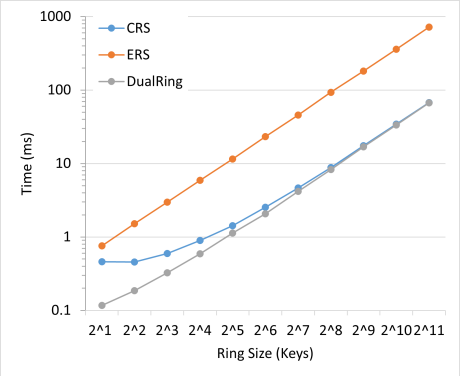


Fig. 4: Time cost for Sign in different ring signature schemes

Referring to Figure 4, we compared the time efficiency of the signing algorithms of our proposed TRS scheme, the ERS scheme with opposite properties, and the prototype DualRing scheme of our algorithm. As the ring size increases from the smallest 2^1 to the largest 2^{11}, the time cost of generating signatures for the three schemes increases exponentially. The time consumption of ERS signature generation is significantly higher than that of the other schemes, while the improved CRS scheme we propose requires only slightly more time than the original DualRing signature.

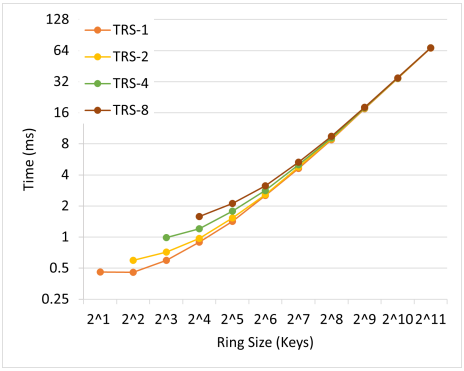


Fig. 5: Time cost for Sign in our TRS for different thresholds

This result is because the ERS scheme uses a Sign-Extend recursive, iterative method, generating standard signatures first and then extending them one by one, and calling the SoK to generate new proofs each time they are extended, which greatly reduces the efficiency of generating ring signatures. In contrast, our scheme is designed without this construction, directly generating ring signatures and introducing additional costs mainly from the initialization hash computation of c_0.

As shown in Figure 5, we compared the time efficiency of our proposed TRS under different threshold settings. As the ring size increases from 2^1 to 2^{11}, the ring size dominates the time cost of signature generation. For signature thresholds of 1, 2, 4, and 8, higher thresholds also bring some extra costs, and the results show that the additional costs only depend on the threshold.

Table 1: Time Cost Introduced by MPyC

Thresholds	1	2	4	8
Time (ms)	0	2.323	3.885	9.3927

It is noteworthy that the TRS algorithm proposed in this study requires collaboration among multiple participants during the generation of signatures. Typically, this collaboration can be achieved through technologies such as multi-party secure computation(MPC) and secret sharing. While this falls outside the scope of our research, it is essential to consider this aspect to align our study with practical scenarios. However, the choice of implementation tools also impacts the testing results of our scheme, hindering our experiments from focusing solely on the contributions of this paper. Given that the time efficiency of multi-party secure computation is generally positively correlated with the number of participants, and the experimental results in Figure 5 also support this trend—larger thresholds result in larger signature sizes—considering the cost of the MPC phase does not alter this outcome. Taking these factors into account, we did not include the additional costs introduced by MPC in the experiments depicted in Figure 5. However, we conducted practical tests on the potential scale of costs introduced by this process.

According to Algorithm 3, MPC only requires joint completion of an additive homomorphic calculation by the signing participants. Based on the statistical results regarding the size of parameter passing, the average size of one parameter passing is 3.661 bytes. To ensure scientific accuracy, experiments were conducted based on a 4-byte data scale, testing the additional costs introduced during the signature generation phase. We utilized the general MPC library, MPyC, for MPC experiments. The results are presented in Table 1, indicating that relative to the signature generation phase, the time cost introduced by MPC is minimal and does not significantly impact the time performance of the signature.

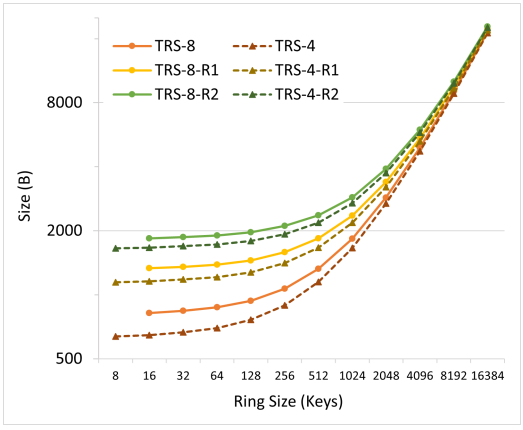
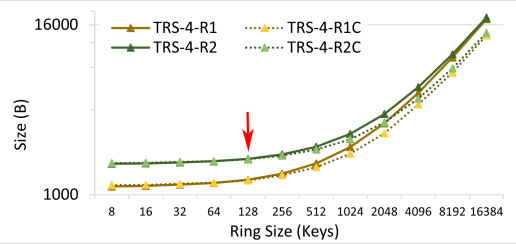


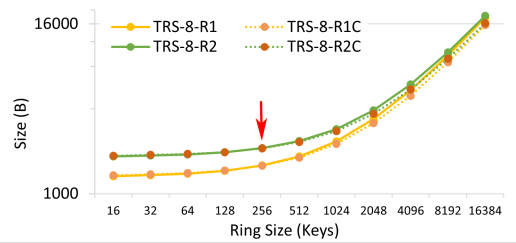
Fig. 6: Signature size for RTRS in different thresholds and different times of Split

The average signature sizes of RTRS are shown in Figure 6. We compared the size of different

signatures under varying ring sizes, including the initially generated TRS, a signature with one and two signers revoked without clipping. Two cases with thresholds of 8 and 4 are performed. The results indicate that, under the same ring size, a larger threshold in the TRS leads to a larger scale, primarily due to the logarithmic correlation between the proof size of NISA and the signature threshold. According to our proposed Split algorithm, each revocation operation by a signer adds information to the signature. Experimental results validate this observation, showing an increase in the signature size with each signer revocation.



(a) Threshold = 4



(b) Threshold = 8

Fig. 7: Signature size for RTRS in different thresholds and different times of Split-and-Clip

Furthermore, we performed maximum clipping on the revoked signatures to examine the changes in signature size. The results are shown in Figure 7, where 7a and 7b illustrate the signature size with and without trimming as the ring size increases, for threshold values of 4 and 8, respectively. The results indicate that for smaller threshold values, the size of withdrawn signatures can be larger than the original signature. However, as the ring size increases, this difference reduces gradually. When the ring is large enough, Split-and-Clip does not incur an extra spatial cost and may even decrease the signature size. For instance, as shown in Figure 7, we observed that the size-optimized critical point is 256 for a threshold of 8 and about 128 for a threshold of 4.

8 Conclusions and Future Works

This paper proposes a novel property of threshold ring signatures, namely revocability. This feature enables signers to revoke their authentication from previously generated signatures without requiring other signers' participation. Moreover, while maintaining anonymity, signers can appropriately reduce the signature's potential anonymity-set size. Revocability eliminates the restrictions of flexibility on threshold ring signatures. It effectively shrinks the signature size when the ring signature's size is large. We present formal models and constructions of CRS (clippable), STRS (splittable), and RTRS (revocable). Furthermore, security proofs are given, and experiments are performed for time and space testing. The results support our conclusions. The revocable threshold ring signature proposed in our work applies to scenarios such as electronic voting, allowing anonymous voters to withdraw their votes, which enhances election fairness non-interactively.

The two directions for the flexibility of threshold ring signatures have been achieved based on the revocability and existing extendability concepts. In future work, exploring the direction of simultaneously satisfying these two properties in one signature scheme is an exciting direction to improve flexibility further. Furthermore, we can explore the general constructions and transformation methods of revocable threshold ring signatures for the proposed schemes and analyze the selection criteria for cryptographic tools suitable for instantiation.

References

[1] Bresson, E., Stern, J. & Szydlo, M. *Threshold Ring Signatures and Applications to Ad-hoc Groups*, 465–480 (2002).

[2] Nojima, R., Tamura, J., Kadobayashi, Y. & Kikuchi, H. *A Storage Efficient Redactable Signature in the Standard Model*, 326–337 (2009).

[3] Brzuska, C. *et al. Redactable Signatures for Tree-Structured Data: Definitions and Constructions*, 87–104 (Berlin, Heidelberg, 2010).

[4] Camenisch, J., Dubovitskaya, M., Haralambiev, K. & Kohlweiss, M. *Composable and Modular Anonymous Credentials: Definitions and Practical Constructions*, 262–288 (2015).

[5] Sanders, O. *Efficient Redactable Signature and Application to Anonymous Credentials*, 628–656 (2020).

[6] Liu, J. K., Wei, V. K. & Wong, D. S. *A Separable Threshold Ring Signature Scheme*, 12–26 (2004).

[7] Liu, D. Y. W., Liu, J. K., Mu, Y., Susilo, W. & Wong, D. S. Revocable Ring Signature. *Journal of Computer Science and Technology* **22**, 785–794 (2007).

[8] Okamoto, T., Tso, R., Yamaguchi, M. & Okamoto, E. A k-out-of-n Ring Signature with Flexible Participation for Signers (2018). URL https://eprint.iacr.org/2018/728. Published: Cryptology ePrint Archive, Paper 2018/728.

[9] Aranha, D. F., Hall-Andersen, M., Nitulescu, A., Pagnin, E. & Yakoubov, S. *Count me in! extendability for threshold ring signatures*, 379–406 (2022).

[10] Avitabile, G., Botta, V. & Fiore, D. Boldyreva, A. & Kolesnikov, V. (eds) *Extendable threshold ring signatures with enhanced anonymity.* (eds Boldyreva, A. & Kolesnikov, V.) *Public-Key Cryptography – PKC 2023*, 281–311 (Springer Nature Switzerland, Cham, 2023).

[11] Rivest, R. L., Shamir, A. & Tauman, Y. *How to Leak a Secret*, 552–565 (2001).

[12] Yuen, T. H., Esgin, M. F., Liu, J. K., Au, M. H. & Ding, Z. *DualRing: generic construction of ring signatures with efficient instantiations*, 251–281 (2021).

[13] Haque, A., Krenn, S., Slamanig, D. & Striecks, C. *Logarithmic-Size (Linkable) Threshold Ring Signatures in the Plain Model*, 437–467 (2022).

[14] Melchor, C. A., Cayrel, P.-L., Gaborit, P. & Laguillaumie, F. A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory* **57**, 4833–4842 (2011).

[15] Gan, Y. A fully adaptively secure threshold signature scheme based on dual-form signatures technology. *Security and Communication Networks* **2021**, 1–11 (2021).

[16] Bootle, J., Cerulli, A., Chaidos, P., Groth, J. & Petit, C. *Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting*, 327–357 (2016).

[17] Bunz, B. *et al. Bulletproofs: Short proofs for confidential transactions and more*, 315–334 (2018).

[18] Bhattacharjee, K. & Das, S. A search for good pseudo-random number generators: Survey and empirical studies. *Computer Science Review* **45**, 100471 (2022).