Traceability for Free: Traceable Ring Signatures Revisited

Xiangyu Liu

CISPA Helmholtz Center for Information Security, xiangyu.liu@cispa.de

Abstract. Linkable Ring Signatures (LRS) allow anonymous signing on behalf of an ad hoc ring, while making any two signatures by the same signer publicly linkable. Traceable Ring Signatures (TRS), introduced by Fujisaki and Suzuki [PKC'07], strengthen linking to tracing: two signatures on different messages additionally reveal the signer's public key. Since cheating almost always means signing two distinct messages, for example two conflicting transactions, traceability turns detection into accountability and makes misbehavior costly. However, TRS remain far less studied and deployed than LRS. We identify two gaps and address both.

First, the security notions inherited from [PKC'07], anonymity, linkability, and exculpability, do not capture attacks specific to tracing. For example, they together do not imply unforgeability, a fundamental requirement for all signature-like primitives. We close this gap by introducing extended linkability and extended exculpability, which reflect the security requirements of TRS more faithfully than their standard counterparts.

Second, all existing TRS schemes trace in $O(n)$ time for a ring of size n, whereas LRS link in $O(1)$. Over a pool of ℓ signatures, pairwise tracing therefore costs $O(\ell^2 n)$ against $O(\ell^2)$ for linking. We design a new TRS framework with $O(1)$ tracing for valid signatures. Instantiated from DDH with Bulletproofs [S&P'18], it yields signatures of $(64 \log n + 544)$ bytes on Curve25519. To our knowledge this is the shortest TRS for $n \geq 12$, and it is on par with state-of-the-art DLog-based LRS schemes. TRS thus gains traceability at essentially no cost over LRS.

Changelog (2026/08/21). This is an updated version of the paper originally titled "Traceable Ring Signatures Revisited: Extended Definitions, $O(1)$ Tracing, and Efficient Log-Size Constructions", posted on 2025/10/02. We make the following main changes.

1. *Security against rogue-key attacks.* The previous version did not consider such attacks, identified by **Claude Opus 5**. We fix this by making the generator $\tilde{g}$ of the Pedersen commitments depend on the ring's public keys.
2. *Corrected concrete signature size (it gets better).* A group element on Curve25519 needs 32 bytes, not 64. Under the correct encoding, our signature size is $(64 \log n + 544)$ bytes rather than $(128 \log n + 736)$, and our construction is the shortest TRS for $n \geq 12$ rather than $n \geq 19$. We update the abstract and Tables accordingly.
3. We rewrite the introduction to streamline the discussion of conceptual limitations in prior TRS work and focus on the essential gaps.

Table of Contents

Traceability for Free: Traceable Ring Signatures Revisited . . . 1
Xiangyu Liu
1 Introduction . . . 3
1.1 Our Contributions . . . 4
1.2 Related Work . . . 5
1.3 Technical Overview . . . 7
1.4 Roadmap . . . 11
2 Preliminaries . . . 11
2.1 Pseudorandom Functions . . . 12
2.2 Zero-Knowledge Proof of Knowledge Arguments . . . 12
3 Traceable Ring Signatures . . . 13
3.1 Security Notions of TRS . . . 14
3.2 Unforgeability and A Variant Definition of Linkability . . . 15
4 Generic Construction of TRS with $O(1)$ Tracing . . . 18
5 Concrete Construction from Bulletproofs in the Random Oracle Model . . . 21
5.1 Instantiation of PRF in the ROM . . . 21
5.2 Instantiation of ZKPoK with Bulletproofs in the ROM . . . 22
5.3 Concrete Signature Size . . . 24
A Hardness Assumptions . . . 31
B Proof of Theorem 2 . . . 31
C Proof of Theorem 5 . . . 32
D Flaws of the Security Proof for Wei et al.'s Scheme [WLB+24, WLB+23] . . . 34
E Insecurity of Qi and Wang's TRS Scheme [QW22] . . . 35

1 Introduction

LRS and TRS. Linkable Ring Signatures (LRS) [LWW04, TWC+04] allow one to sign anonymously on behalf of an ad hoc group, called a *ring*, while providing linkability: any two signatures produced by the same signer under a common *event label* $\mathsf{e} \in \{0,1\}^*$ are publicly recognizable as linked by algorithm Link. The ring is assembled on the fly without any setup, group manager, or cooperation from other group members, and the event label delimits the scope of linking, for example a voting round or a payment epoch. Since double-signing is then publicly detectable, LRS have found wide applications in e-voting [CLW08], e-cash [CHL05, TW05], ad hoc network authentication [LWW04], anonymous off-line coupon services [Fuj11] and private payments [LRR+19]. They are also deployed at scale, most notably in Monero's RingCT protocol [YSL+20], where the link algorithm prevents double-spending.

Linkability, however, delivers detection without accountability. A signer who double-votes or double-spends is exposed as having done so, yet remains hidden inside the ring, so the offending signature can be discarded but no one can be penalized. Traceable Ring Signatures (TRS), introduced by Fujisaki and Suzuki [FS07], close this gap by replacing Link with algorithm Trace and lifting linkability to *traceability*: two signatures by the same signer under the same event label remain linked, and if they are on *different* messages, Trace additionally outputs the signer's public key.

Traceability is preferable to linkability in practice, since genuine misbehavior almost always yields two *distinct* messages: a double-spend is necessarily two distinct transactions (e.g., differing in recipient, amount or fee), and a double-vote involves two distinct ballots, or at least two fresh serial numbers or timestamps even when the same choice is encoded. Signing the same message twice, by contrast, is usually regarded as a mere duplicate, and the system simply discards one copy. Therefore, traceability reveals the signer precisely when a real misbehavior happens, which is required wherever detection alone is insufficient and offenders must be held accountable.

Revisiting TRS: Gaps in Efficiency and Security. Despite offering the strictly stronger guarantee of traceability, TRS remain far less studied than LRS, which, despite their weaker linkability guarantee, are already widely deployed in practice [Noe15, LRR+19, YSL+20, TSS+18, KWT25]. We identify two gaps behind this discrepancy: one in efficiency, the other in how TRS security is formalized.

The efficiency gap. In most LRS schemes, Link is essentially free: it only checks whether two signatures σ and σ' carry the same (or sufficiently close) *pseudo-identity*, a linking tag determined solely by the event label e and the signer's secret key, so Link runs in $O(1)$ time. By contrast, all existing TRS constructions compute Trace in $O(n)$ time, where n is the ring size, since Trace first computes two length-n sequences from the two signatures, and then returns the index where the two sequences agree. Constructing a TRS scheme with $O(1)$ tracing remains an open problem, and solving it would make TRS as practical as LRS while retaining strictly stronger accountability.

One might think this gap is moot, since verifying a signature already costs $O(n)$ before it is even worth tracing. This overlooks that verification and tracing scale with the size of the *signature pool*, not just the ring, and are typically handled by distinct roles running at very different frequencies. For example, in e-voting, a validation unit verifies each incoming ballot, while an oversight unit later scans the entire pool via Link or Trace to catch double-voting. Over a pool of ℓ signatures—typically far larger than n—verification is invoked only ℓ times, whereas tracing requires $\ell(\ell-1)$ pairwise checks. This quadratic blow-up leaves the oversight unit's cost dominated entirely by Link or Trace, and has explained why LRS's cheap $O(1)$ linking is more attractive for large-scale deployments than TRS's $O(n)$ tracing.

The security gap. TRS security is also less settled than that of LRS, whose notions have been progressively sharpened since Liu et al.'s original work [LWW04] by a line of follow-up results [TWC+04, LW06, BKP20, BGK+21, XLAZ24, CLXZ25]. The TRS literature instead settles on three properties:

- *Linkability.* An adversary corrupting up to ℓ parties cannot produce $\ell+1$ pairwise unlinked signatures.
- *Exculpability* (aka non-slanderability). An adversary cannot forge two signatures, one of which may come from the signing oracle, that trace to an honest user's public key.

– *Anonymity.* A signature does not reveal its signer's identity.

One property missing from this list is *unforgeability*, the baseline requirement for all signature-like primitives. Fujisaki and Suzuki [FS07] proved that linkability and exculpability together imply unforgeability, in analogy with the corresponding argument for LRS [ASY06]. The high-level idea is that any forged signature σ^* should be either unlinked from every existing signature σ, which breaks linkability, or linked to some σ and hence traced to a public key, which breaks exculpability. This argument is sound for LRS, but it does not hold in TRS, since Trace may return "linked" without outputting any public key at all. We observe that this gap stems from the insufficient formalization of linkability and exculpability in TRS. As currently defined, they overlook attacks specific to the trace-versus-link distinction that have no counterpart in LRS. Closing it requires reformulating the security notions, to actually capture the security requirements arising in practice.

1.1 Our Contributions

In this work, we revisit Traceable Ring Signatures (TRS), identify several gaps in the security notions, and provide a new framework with efficient trace, making TRS as practical as the more widely studied linkable ring signatures but with a stronger trace functionality. We summarize our contributions as follows.

1. We identify that linkability and exculpability defined by Fujisaki and Suzuki [FS07] and followed by the literature are insufficient to capture the real-world threats of TRS, for example, they together do not imply unforgeability as claimed in [FS07]. We re-formalize the security notions, and introduce extended linkability and extended exculpability to fix this gap.
   – Extended linkability: the adversary can neither (1) forge $\ell + 1$ pairwise unlinked signatures from ℓ corruptions, nor (2) (the extended part) forge two signatures on different messages that are linked but fail to trace to a corrupted pk.
   – *Extended exculpability*: the adversary can neither (1) forge two signatures that trace to an honest pk, nor (2) (the extended part) forge a signature that is linked to an existing signature of an honest user.

   We illustrate different security notions and their relations in Fig. 1.
2. We design a new framework of TRS from PseudoRandom Functions (PRF) and Zero-Knowledge Proof of Knowledge arguments (ZKPoK) that supports $O(1)$ tracing, assuming both signatures are valid. By contrast, all previous constructions incur a tracing complexity of $O(n)$. Our framework substantially improves the concrete efficiency of tracing—independent of the specific instantiation—and renders TRS as practical and efficient as LRS.

   Our new framework also supports event-orbited tracing, aligned with the more generalized event-orbited linking in LRS [TWC+04, LW06], i.e., Trace functions as long as the two signatures are under the same event label. By contrast, all previous TRS schemes have a stricter requirement that two signatures should be associated with the same ring.
3. We instantiate our generic framework from the DDH assumption and Bulletproofs [BBB+18] in the random oracle model. The scheme achieves log-size signatures and remains fully compatible with most existing (DLog-based) systems, showing that TRS can be as efficient as LRS while additionally providing traceability. See Table 1 for a detailed comparison with some state-of-the-art (DLog-based) LRS schemes and all other TRS schemes.

On the Necessity of Efficient Tracing. As discussed above, an $O(1)$ Trace matters for more than shaving a constant off a single comparison. Real deployments must trace *every pair* among a pool of ℓ signatures, so an $O(n)$-per-call Trace costs $O(\ell^2 n)$ in total, whereas an $O(1)$-per-call Trace costs $O(\ell^2)$, and it can further collapse to $O(\ell)$ with $O(\ell)$ memory by hashing pseudo-identities into buckets and flagging collisions.

Our framework in Sec. 4 achieves exactly this. Given σ and σ', Trace first checks whether their pseudo-identities match, exactly as Link does in most LRS schemes. If they match and the two messages differ, it performs additionally a scalar multiplication on the pseudo-identities and outputs the result as the traced

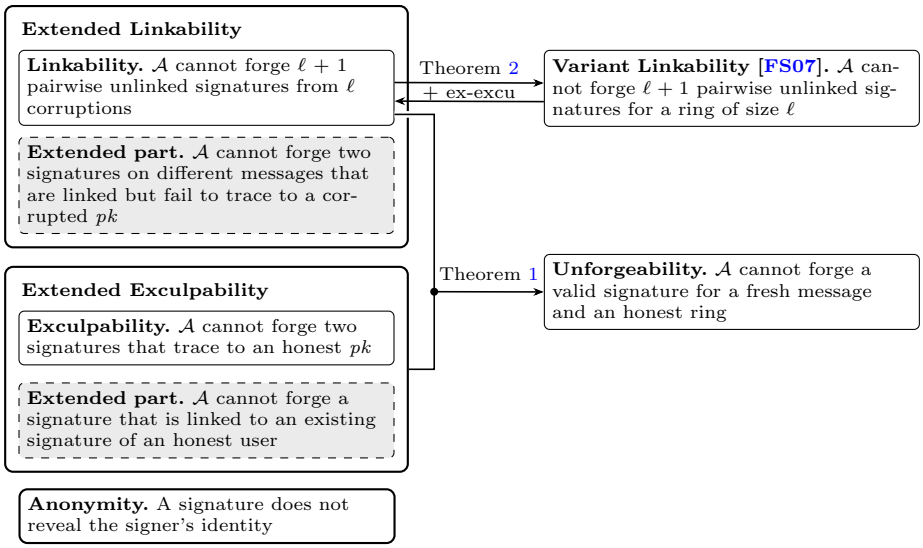

Fig. 1. Security notions of traceable ring signatures (TRS) and their relations. The *extended parts* introduced in this work are shaded. Theorem 1 shows that linkability together with extended exculpability implies unforgeability, repairing the proof of [FS07]. Theorem 2 relates our linkability to the variant notion adopted in prior work.

public key. Existing TRS constructions [Fuj11, GDW20, SZ21, BM19, FLWL20, FLL+21, WLB+24, YLGT23, KSD+24], by contrast, all follow the original design by Fujisaki and Suzuki [FS07] and incur $O(n)$ tracing, where Trace recomputes and compares two length-n sequences per call. Bucketing helps only so far there: even with $O(\ell n)$ memory, each signature's length-n sequence must still be recomputed, leaving a total time cost of $O(\ell n)$. This gap is significant in settings where the pool size ℓ and the ring size n are both large.

Notes on Table 1. We do not consider TRS schemes where a trusted authority is needed for issuing keys or trace (e.g., identity-based TRS [ALSY13, PGLZ21, LHH+23b, YW25], certificate-less TRS [LHH+23a], TRS schemes in [GW18, LHC23, XZC+24, ZCW+24], auditable privacy-preserving cryptocurrency [LWC+19], etc.). We do not compare the schemes that are pointed out to be incomplete or insecure, including:
- [BM19]: a Syndrome Decoding-based TRS scheme, of which insecurity was pointed out by Feng et al. [FLWL20, FLL+21].
- [BL16]: a k-time TRS scheme, of which the insecurity was pointed out by Choi et al. [CLXZ25].
- [QW22]: a code-based TRS scheme, of which the insecurity was pointed out by us in Appendix E.

The assumptions include: Q-DDHI: the Q-Decisional Diffie-Hellman Inversion assumption; SXDH: the Symmetric External Diffie-Hellman assumption; (Q_1, Q_2)-DL: the (Q_1, Q_2)-Discrete Logarithm assumption; Q-SDH: the Q-Strong Diffie-Hellman assumption; DH-family: the Subgroup Decision assumption, the Q-Strong Diffie-Hellman assumption and the Pseudo-Random DDHI Assumption; (M-)LWE: the (Module) Learning With Errors assumption; (M-)SIS: the (Module) Short Integer Solution assumption. Symmetric-key primitives: LowMC [ARS+15], symmetric-key-based accumulators [DRS18]; CSIDH: the Commutative Supersingular Isogeny Diffie Hellman assumption.

1.2 Related Work

(Linkable) Ring Signatures. Ring signatures [RST01] allow a user to sign a message on behalf of an ad-hoc group without revealing the identity. Classical ring signatures are unlinkable, meaning that it is impossible to determine whether two signatures were generated by the same signer. Liu et al. [LWW04] introduced Linkable Ring Signatures (LRS), where a link algorithm can publicly determine if two signatures originate from the same signer or not. With this linkability, LRS has found extensive applications in e-voting [CLW08],

Table 1. Comparison with LRS schemes (the upper block) and TRS schemes (the lower block). Here n denotes the ring size, CRS denotes the Common Reference String model, and AGM denotes the Algebraic Group Model. Our construction attains the same $O(1)$ link/trace complexity as in LRS while additionally providing traceability. Concrete sizes are reported only for DLog-based schemes. [a] In one-time TRS [SZ21], each key pair can be used to sign at most one message. [b] We point out several flaws in the security proof of [WLB+24] in Appendix D.

Scheme	Link/Trace Complexity	Model	Assumption	Asymptotic Size	Concrete Size
Linkable ring signatures (linkability only)					
LLW04 [LWW04]	$O(1)$	ROM	DDH	$O(n)$	$\lvert\mathbb{G}\rvert + (n+1)\cdot\lvert\mathbb{Z}_q\rvert$
Omniring [LRR+19]	$O(1)$	ROM	DDH	$O(\log n)$	$(2\log(n+2)+4)\cdot\lvert\mathbb{G}\rvert + 5\cdot\lvert\mathbb{Z}_q\rvert$
RingCT 3.0 [YSL+20]	$O(1)$	ROM	Q-DDHI	$O(\log n)$	$(2\log n+7)\cdot\lvert\mathbb{G}\rvert + 7\cdot\lvert\mathbb{Z}_q\rvert$
LLRing-DL [HC24]	$O(1)$	ROM	DLog	$O(\log n)$	$(4\log n+9)\cdot\lvert\mathbb{G}\rvert + 7\cdot\lvert\mathbb{Z}_q\rvert$
LLRing-P [HC24]	$O(1)$	ROM	SXDH	$O(\log n)$	$(6\log n+10)\cdot\lvert\mathbb{G}_T\rvert + 11\cdot\lvert\mathbb{Z}_q\rvert$
XTA+25 [XTA+25]	$O(1)$	ROM, CRS, AGM	(Q_1,Q_2)-DL, Q-SDH	$O(1)$	$10\cdot\lvert\mathbb{G}_1\rvert + 5\cdot\lvert\mathbb{G}_2\rvert + \lvert\mathbb{Z}_q\rvert$
Traceable ring signatures (linkability and traceability)					
This work	$O(1)$	ROM	DDH	$O(\log n)$	$(2\log n+6)\cdot\lvert\mathbb{G}\rvert + 11\cdot\lvert\mathbb{Z}_q\rvert$
FS07 [FS07]	$O(n)$	ROM	DDH	$O(n)$	
Fuj11 [Fuj11]	$O(n)$	CRS	DH-family	$O(\sqrt{n})$	
SZ21 [SZ21] (one-time TRS[a])	$O(n)$	ROM	none	$O(n)$	
FLWL20 [FLWL20, FLL+21]	$O(n)$	(Q)ROM	LWE, SIS	$O(\log n)$	
FLL+21 [FLL+21]	$O(n)$	(Q)ROM	Symmetric-key primitives	$O(\log n)$	
YLGT23 [YLGT23]	$O(n)$	ROM	MSIS, MLWE	$O(n)$	
WLB+24-ISO [WLB+24][b]	$O(n)$	ROM	CSIDH	$O(\log n)$	
WLB+24-LAT [WLB+24][b]	$O(n)$	ROM	MSIS, MLWE	$O(\log n)$	
KSD+24 [KSD+24]	$O(n)$	Plain	NIWI, VRF, PKE and SPB	$O(\log n)$	

e-cash [TW05], ad-hoc network authentication [LWW04], private payment [LRR+19], etc. Recent work of LRS therefore focuses on log-size signatures [GK15, LRR+19, YSL+20, YEL+21, BEHM22, HC24] (or even constant-size in the CRS setting [XTA+25]), and post-quantum instantiations [ESS+19, EZS+19, BM18, BKP20].

Traceable Ring Signatures. In LRS, the identity of a malicious user is hidden even if many linked signatures from it are observed. To mitigate "excessive" anonymity of LRS, Fujisaki and Suzuki [FS07] introduced Traceable Ring Signatures (TRS) where the link algorithm is enhanced to a trace algorithm such that, two linked signatures on different messages will expose the signer's identity. TRS have proven to be a powerful tool to enhance the fairness of robust voting systems, as they not only prevent double-voting but also enable the identification and accountability of dishonest voters.

Log-Size TRS. Early constructions of LRS and TRS have a signature size linear in the ring size. Following the step of log-size LRS [LRR+19, YSL+20, HC24], recent works have explored log-size TRS constructions. Branco and Mateus [BM19] proposed the first code-based TRS scheme from a 1-out-of-n proof [CDS94] for the GStern's protocol [BM18]. However, the scheme was later pointed out to be insecure by Feng et al. [FLWL20, FLL+21], since the OR-composition method [CDS94] is not applicable to the (G)Stern protocols. In [FLL+21], Feng et al. proposed a general framework of TRS from PRF, collision-resistant hashes, and ZKPoK, and provided a log-size instantiation from lattices and another instantiation from symmetric-key primitives. Wei et al. [WLB+24] constructed two log-size TRS schemes from group actions and instantiated them with isogenies and lattices, respectively. Recently, Khuc et al. [KSD+24] presented the first generic construction of traceable ring signature schemes without (quantum) random oracles from non-interactive witness-indistinguishable proofs (NIWI), verifiable random functions (VRF), somewhere perfectly binding hash functions (SPB). However, there is no discussion on implementations.

Other Variants of TRS. There are other variants of TRS, for example, traceable extendable threshold ring signatures [ABF25], report and trace ring signatures [FQ21], etc. We do not consider those variants in this work.

1.3 Technical Overview

In this subsection we give a brief technical overview.

Revisiting Traceable Ring Signatures. In [FS07], the trace algorithm Trace is defined on a tag $L = (\mathsf{issue}, PK_R)$. Here $\mathsf{issue} \in \{0,1\}^*$ refers to a signing issue (the same as the event label e in this paper) and PK_R denotes the public key list of the ring. Consequently, $\mathsf{Trace}(L, \sigma, \sigma')$ functions only if σ and σ' are related to the same issue and ring. To make the definition more general, we first extend the syntax by defining Trace on the event label $\mathsf{e} \in \{0,1\}^*$ only, as the work already done in linkable ring signatures [TWC+04, LASZ13, CLXZ25]. By including PK_R into e, it covers the definition in [FS07].

Except for correctness, [FS07] (and most the follow-up works [Fuj11, BM19, FLWL20, FLL+21, YLGT23, WLB+24]) considered the following three properties for TRS.

- **(Variant-)Linkability [FS07].** The adversary cannot forge $\ell+1$ signatures for a ring of size ℓ that are pairwise independent (unlinked). We refer to the notion in [FS07] as variant linkability, to differ from the (stronger) linkability notion adopted in this work.
- **Exculpability.** The adversary cannot forge two signatures, of which one can be from the signing oracle, that trace to an honest user's public key.
- **Anonymity.** The signature does not reveal the signer's identity, except for the information leaked by Trace.

One missing property here is unforgeability, the basic security requirement for all signature-like primitives. Fujisaki and Suzuki [FS07] claimed that linkability and exculpability together imply unforgeability. The high-level proof idea is as follows. Let $(R^*, \mathsf{e}^* = (\mathsf{issue}^*, R^*), \mathsf{m}^*, \sigma^*)$ be the adversary $\mathcal{A}$'s final output against the unforgeability, where $R^* = \{n_1, ..., n_\ell\}$, and all users in R^* are uncorrupted. Consider ℓ messages $(\mathsf{m}_1 \neq \mathsf{m}^*, ..., \mathsf{m}_\ell \neq \mathsf{m}^*)$ and ℓ signatures $((R_1, \mathsf{m}_1, \sigma_1), ..., (R_\ell, \mathsf{m}_\ell, \sigma_\ell))$ signed by users n_1, ..., n_ℓ, respectively.[1] Then [FS07] argued that if all $\ell+1$ signatures are pairwise independent, then it contradicts linkability. Otherwise, by the property of Trace we have $\mathsf{Trace}(\mathsf{e}^*, \sigma_j, \sigma^*) = pk$ for some $j \in [\ell]$ since $\mathsf{m}_i \neq \mathsf{m}^*$, which contradicts exculpability.

However, the argument "$\mathsf{Trace}(\mathsf{e}^*, \sigma_i, \sigma^*)$ always outputs a public key if $\mathsf{m}_i \neq \mathsf{m}$" in the second case is flawed, since it holds only if both signatures are honestly generated, as stated by correctness. Recall that σ^* is a forgery by the adversary $\mathcal{A}$. Therefore, the aforementioned statement cannot be applied here, and it may be feasible for $\mathcal{A}$ to generate an adversarial signature σ^* on message $\mathsf{m}^* \neq \mathsf{m}_i$ that is linked to σ_i, but failed to trace to the i-th user's public key. In this case, $\mathcal{A}$ breaks neither the linkability nor the exculpability.

Such a proof gap is due to the incomplete definition of exculpability in [FS07]. Namely, exculpability captures only the infeasibility of $\mathcal{A}$ on forging a signature that together with an existing signature trace to one honest user. However, it overlooks the case that the two signatures are linked but not traced to any public key—another threatening attack in practice that may cause the signature from the honest user uncounted in the application systems, for example, e-voting. To address this, we consider the following extended exculpability:

- **Extended exculpability.** An adversary can neither (1) forge two signatures (one can be from the signing oracle) that traced to an honest user ($\mathsf{win}_3 \wedge \mathsf{win}_4$ in Fig. 3), nor (2) (the extended part) forge a signature that is linked to an existing signature of an honest user ($\mathsf{win}_5 \wedge \mathsf{win}_6$ in Fig. 3). Note that the second case may not necessarily output a traced public key.

[1] How the ℓ signatures are generated is not specified in [FS07]. Considering the following argument in the proof, we think they are generated by invoking Sign honestly.

We also observe that linkability in [FS07] was derived from the linkability defined for LRS [LWW04, BKP20, XLAZ24]. However, since TRS strictly extended LRS (tracing signatures on different messages to a public key), it is necessary to consider attacks targeting this extended functionality, i.e., attacks that cause the tracing to fail. Note that, as discussed above, such attacks are not captured by correctness, which guarantees trace validity only for honestly generated key pairs and signatures. Therefore, we define the following extended linkability:

– **Extended linkability.** An adversary can neither (1) forge $\ell + 1$ pairwise independent (unlinked) signatures from at most ℓ corruptions (win_4 in Fig. 3), nor (2) (the extended part) forge two signatures on different messages that are linked but fail to trace to a corrupted public key ($\mathsf{win}_5 \wedge \mathsf{win}_6 \wedge \mathsf{win}_7$ in Fig. 3).

Independent of the extended part, we also revise the notion of (standard) linkability, which is strictly stronger than that defined in [FS07]. Compared to previous notions, our revision prevents the adversary from artificially increasing its advantage by including extra honest users into the ring. We prove that, conditioned on *extended* exculpability, the variant linkability implies the linkability in this paper. Although a provably secure TRS scheme should satisfy both variant linkability and extended exculpability—and therefore the (standard) linkability is implied—we still adopt this revised notion to provide a clearer understanding of the distinct security properties of TRS and their dependence on the underlying hardness assumptions.

We emphasize that extended linkability and extended exculpability are strictly stronger than their standard counterparts in the literature. To see this, we adapt any secure TRS scheme, say the generic construction of Fig. 5, into a scheme $\mathsf{TRS}^\dagger$ that additionally offers a *phantom* signing mode. A phantom signature is an ordinary signature prefixed with a dummy string 1^λ. Ver accepts it if and only if it accepts the underlying ordinary signature, and $\mathsf{Trace}(\sigma, \sigma')$ directly returns linked whenever one of its inputs is a phantom signature. Honest signers never invoke the phantom mode, so correctness and anonymity are inherited from the underlying scheme. Unforgeability is inherited as well, since a phantom signature still embeds a genuine one. Standard linkability is inherited because a phantom signature is linked to every signature under the same event label and therefore never belongs to a set of pairwise independent signatures, and standard exculpability is inherited because Trace never outputs a public key once a phantom signature is involved. Nevertheless, $\mathsf{TRS}^\dagger$ satisfies neither extended notion: an adversary corrupting a single user can either (i) publish a phantom signature that is linked to an honestly generated signature, breaking extended exculpability, or (ii) publish a phantom signature together with an ordinary one of its own on a different message, which are linked yet trace to no corrupted public key, breaking extended linkability.

The TRS Scheme in [FS07]. The TRS scheme in [FS07] is based on the DDH (and hence the DLog) assumption. Let $\mathbb{G}$ be a cyclic group of prime order q with g a generator. A pair of public key and secret key is of the form $(X = g^x, x) \in (\mathbb{G}, \mathbb{Z}_q)$. Let $PK_R = (X_1, ..., X_n)$ be the public key list of ring $R = \{1, ..., n\}$ and $\mathsf{e} = (\mathsf{issue}, PK_R)$. Let $H, H' : \{0,1\}^* \to \mathbb{G}$ be two hash function that modeled as random oracles, and $g_\mathsf{e} = H(\mathsf{e})$. A signature on m generated by the δ-th user holding x_δ is of the form

$$\sigma = (A_1, ..., A_n, \pi),$$

where the ordered sequence $(A_1, ..., A_n)$ satisfies that

$$A_{i+1}/A_i = H'(\mathsf{e}, \mathsf{m}),\ \forall i \in [n-1],$$

and π is a zero-knowledge proof such that

$$\exists \delta \in R\ s.t.\ \log_g X_\delta = \log_{g_\mathsf{e}} A_\delta.$$

In fact, it is sufficient to contain only A_1 in the signature since $\{A_i\}_{2 \le i \le n}$ can be deduced from A_1, e, and m. Here we assume they are given explicitly in the signature for better analyzing the structure.

Now let us take a closer look at the construction. At first, to achieve linkability as in LRS (i.e., two signatures from the same user will be linked), the signature contains a pseudo-identity $A_\delta = g_\mathsf{e}^{x_\delta}$ which is

determined only by the secret key (which is linked to the user's identity) and the event label $\mathsf{e} = (\mathsf{issue}, PK_R)$. Therefore, two signatures will be linked if they contain the same pseudo-identity. To hide δ, the index of the signer among the ring, the signer generates other $n-1$ pseudo-identities $\{A_i\}_{i\neq\delta}$ according to the recurrence relation $A_{i+1}/A_i = H'(\mathsf{e}, \mathsf{m})$, which is determined by the event and the message. At last, the signer generates a zero-knowledge proof to show that there is one pseudo-identity A_δ among $A_1, ..., A_n$ that is well-formed. For a fixed event label e, the functionality of TRS is guaranteed as follows.

- If the signer signs two signatures on the same message, then $A_1, ..., A_n$ will be identical in both signatures, and the trace algorithm returns linked accordingly.
- If the signer signs two signatures $(A_1, ..., A_n, \pi)$ and $(A'_1, ..., A'_n, \pi')$ on different messages, then the "real" pseudo-identity $A_\delta = g_{\mathsf{e}}^{x_\delta}$ remains the same in both signatures but the recurrence relations are different. Therefore, among $(A_1, ..., A_n)$ and $(A'_1, ..., A'_n)$ there will be a unique index δ that $A_\delta = A'_\delta$, and the trace algorithm returns the public key pk_δ.

The idea in [FS07] was later followed by other subsequent works to construct TRS from a variety of assumptions. In [FLL+21], Feng et al. abstracted [FS07]'s scheme as a generic construction from Pseudo-Random Functions (PRF) and Zero-Knowledge Proof of Knowledge arguments (ZKPoK). Let α be a public random string and sk be the secret key of a pseudorandom function $F_{sk}(\cdot)$ whose image space forms an additive group. Given public key list $(pk_1 = F_{sk_1}(\alpha), ..., pk_n = F_{sk_n}(\alpha))$ and event label e, a TRS signature is of the form

$$\sigma = (A_1, ..., A_n, \pi),$$

where the ordered sequence $(A_1, ..., A_n)$ satisfies that $A_{i+1} - A_i = H'(\mathsf{e}, \mathsf{m}),\ \forall i \in [n-1]$, and π is a zero-knowledge proof for the following language:

$$\mathcal{L}_{[\mathsf{FLL+21}]} : \{(pk_1, ..., pk_n, A_1, ..., A_n) \mid \exists(\delta, sk_\delta)\ s.t.\ \delta \in [n] \wedge pk_\delta = F_{sk_\delta}(\alpha) \wedge A_\delta = F_{sk_\delta}(\mathsf{e})\}.$$

However, there are two inherent shortcomings of this framework. First, the aforementioned construction does not support event-orbited tracing. Second and more importantly, Trace runs in $O(n)$ time, since it compares two sequences of length n. Though verification already costs $O(n)$, as discussed earlier, it is performed once per signature whereas tracing is performed once per pair, so reducing the tracing cost remains well motivated.

New Framework of TRS with $O(1)$ Tracing. Inspired by the LRS schemes [LWW04, CLXZ25] and the TRS schemes [FS07, FLL+21], our starting point is the following generic construction of LRS from PRF and ZKPoK:

Given a public key list $(pk_1 = F_{sk_1}(\alpha), ..., pk_n = F_{sk_n}(\alpha))$ and an event label e, an LRS signature is $\sigma = (pid, \pi)$, where $pid = F_{sk_\delta}(\mathsf{e})$, and π is a zero-knowledge proof for the following language:

$$\mathcal{L}_{\mathsf{LRS}} : \{(pk_1, ..., pk_n, pid) \mid \exists(\delta, sk_\delta)\ s.t.\ \delta \in [n] \wedge pk_\delta = F_{sk_\delta}(\alpha) \wedge pid = F_{sk_\delta}(\mathsf{e})\}.$$

Now we try to enhance the above construction to a TRS scheme. Traceability requires that, given event label e and two signatures σ, σ',

1. if σ and σ' are associated with the same message m, then $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma')$ just returns linked and no information about pk is leaked, and
2. if σ and σ' are associated with different messages m and m', then $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma')$ reveals pk.

Our key insight here is that the above two properties of Trace have similarities in spirit with a secret sharing scheme [Sha79]. Regarding pk as the secret and the signature σ on m as a share of pk, then

1. one share σ (i.e., the m-th share of pk) leaks no information about pk, and
2. two shares σ and σ' (i.e., the m-th and the m'-th shares of pk) together reveal the secret information pk. Meanwhile, the reconstruction of pk is $O(1)$.

Holding the above observation in mind, we borrow the trace trick from the compact e-cash scheme of Camenisch, Hohenberger, and Lysyanskaya [CHL05], to hide the public key information inside the pseudo-identity. Namely, we design the m-th share of pk as $(u + \mathsf{m} \cdot pk)$, where u is a random masking element. Combined with the above generic construction of LRS, we propose the following new framework of TRS scheme with $O(1)$ trace:

Given public key list $(pk_1 = F_{sk_1}(\alpha), ..., pk_n = F_{sk_n}(\alpha))$ and event label e, a TRS signature is $\sigma = (pid_1, pid_2, \pi)$, where

$$pid_1 = F_{sk_\delta}(\mathsf{e}||1),\ pid_2 = F_{sk_\delta}(\mathsf{e}||2) + \mathsf{m} \cdot pk_\delta,$$

and π is a zero-knowledge proof for the following language:

$$\begin{aligned}\mathcal{L}_{\mathsf{TRS}} : \{(pk_1, ..., pk_n, pid_1, pid_2) \mid \exists(\delta, sk_\delta)\ s.t.\ \delta \in [n]\ \wedge\ pk_\delta = F_{sk_\delta}(\alpha) \\ \wedge\ pid_1 = F_{sk_\delta}(\mathsf{e}||1)\ \wedge\ pid_2 = F_{sk_\delta}(\mathsf{e}||2) + \mathsf{m} \cdot pk_\delta\}.\end{aligned}$$

We briefly analyze the correctness and security of the above framework.

- **Linkability (on the signatures w.r.t. the same message).** The trace algorithm will return linked due to the identical pid_1 and pid_2 in both signatures.
- **Traceability (on the signatures w.r.t. different messages).** The trace algorithm will return $(\mathsf{linked}, pk_\delta)$ due to the identical pid_1 and the reconstruction of pk_δ from $pid_2 = F_{sk_\delta}(\mathsf{e}||2) + \mathsf{m} \cdot pk_\delta$ and $pid_2' = F_{sk_\delta}(\mathsf{e}||2) + \mathsf{m}' \cdot pk_\delta$. Moreover, the reconstruction of pk costs $O(1)$ time.
- **Anonymity.** This is guaranteed by the pseudorandomness of PRF and the zero-knowledge of ZKPoK.
- **Traceability and exculpability.** They are (mainly) guaranteed by the (strong) simulation extractability of ZKPoK.

Log-Size Instantiation from the Bulletproofs. Note that the signature size is dominated by the size of π, whose core component is a set membership proof showing the knowledge of a secret key among n public keys. The seminal work of Bulletproofs [BBB+18] provides a log-size (zero-knowledge) range proof scheme from the DLog assumption. Combined with the DDH-based instantiation of PRF, we obtain a concrete log-size TRS scheme with $O(1)$ tracing in the random oracle model.

In more detail, let $H : \{0,1\}^* \to \mathbb{G}$ be a hash function modeled as a random oracle. We instantiate PRF $F_{sk}(z)$ with secret key $sk \in \mathbb{Z}_q$ and input $z \in \{0,1\}^*$ as follows:

$$F_{sk}(z) = (H(z))^{sk}.$$

Let $(pk, sk) = (X = g^x, x)$ be a pair of public and secret keys, and $(g_{\mathsf{e},1}, g_{\mathsf{e},2}) = (H(\mathsf{e}||1), H(\mathsf{e}||2)) \in \mathbb{G}^2$ be the hash outputs of an event label $\mathsf{e} \in \{0,1\}^*$. By the DDH assumption, $g_{\mathsf{e},1}^x$ and $g_{\mathsf{e},2}^x$ are pseudorandom given public key g^x and public inputs $g_{\mathsf{e},1}$, $g_{\mathsf{e},2}$.

Let $PK_R = (X_1, ..., X_n) \in \mathbb{G}^n$ be a set of public keys, and $\mathsf{m} \in \mathbb{Z}_q$ be the message. Instantiating the PRF scheme as above, the language of the ZKPoK scheme becomes

$$\begin{aligned}\mathcal{L}_{\mathsf{DL\text{-}TRS}} := \{ins = (X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2) \mid \exists wit = (\delta, x)\ s.t.\ \delta \in [n]\ \wedge\ X_\delta = g^x \\ \wedge\ pid_1 = g_{\mathsf{e},1}^x\ \wedge\ pid_2 = g_{\mathsf{e},2}^x \cdot g^{x \cdot \mathsf{m}}\}.\end{aligned}$$

Following [BBB+18], the set membership proof for $\delta \in [n]$ can be achieved by proving the knowledge of $\mathbf{a}_L$ such that

$$\langle \mathbf{a}_L, \mathbf{1} \rangle = 1\ \wedge\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}\ \wedge\ \mathbf{a}_R = \mathbf{a}_L - \mathbf{1},$$

where $\mathbf{a}_L = (a_1, ..., a_n) \in \{0,1\}^n$, $a_\delta = 1$ and $a_{i \neq \delta} = 0$. Meanwhile, given $C = g^x \tilde{g}^{r_C}$ as a commitment of $pk = g^x$, the well-formedness of $pid_1 = g_{\mathsf{e},1}^x$ and $pid_2 = g_{\mathsf{e},2}^x \cdot g^{x \cdot \mathsf{m}}$ can be achieved via a Schnorr-like Sigma protocol [Sch90] and the Fiat-Shamir transform [FS87].

Avoiding Rogue-key Attacks. As described so far, the construction admits a rogue-key attack against extended linkability. An adversary can sample $x, d \leftarrow \mathbb{Z}_q$, and register a public key $X_\delta = g^x \cdot \tilde{g}^d$ for which it knows a representation in the basis $(g, \tilde{g})$. Observe that

$$C = g^x \cdot \tilde{g}^{r_C} = (g^x \tilde{g}^d) \cdot \tilde{g}^{r_C - d}.$$

Using the opening $(X_\delta, r_C - d)$, the adversary proves that C commits to a public key X_δ in PK_R; and using (x, r_C), it proves knowledge of x and the well-formedness of pid_1, pid_2 w.r.t. g^x. Both statements are true, yet together they no longer imply that the committed key equals g^x. The adversary can thus forge two valid signatures on distinct messages $\mathsf{m} \neq \mathsf{m}'$ such that

$$\begin{cases} pid_1 = H(\mathsf{e}||1)^x, \ \wedge \ pid_2 = H(\mathsf{e}||2)^x + (g^x)^{\mathsf{m}}, \\ pid_1 = H(\mathsf{e}||1)^x, \ \wedge \ pid_2 = H(\mathsf{e}||2)^x + (g^x)^{\mathsf{m}'}. \end{cases}$$

which are linked but traced to g^x rather than the registered public key X_δ, hence breaking extended linkability.

This does not contradict the binding property of Pedersen commitments, which is computational: finding two openings of C is as hard as DLog. The point is that the adversary never has to find them. Since $\tilde{g}$ is a fixed element, the adversary can build its public key so that one opening is known by construction.

To fix it, we make the second generator depend on the ring, setting $\tilde{g} := H'(PK_R)$ with H' another random oracle. Two openings of the same C then yield a representation $X_\delta = g^x \tilde{g}^e$ with $e \neq 0$ of a key fixed before $\tilde{g}$ was sampled, and by forking lemma this helps to extract a DLog solution embedded in $\tilde{g}$.

Taking all together, the resulting TRS scheme has a signature size $((2\log n + 6) \cdot |\mathbb{G}| + 11 \cdot |\mathbb{Z}_q|)$. Instantiating with Curve25519 which requires 32 bytes for both an element in $\mathbb{Z}_q$ and a group element in $\mathbb{G}$, the concrete signature size is $(64 \cdot \log n + 544)$ bytes. See Table 2 for a detailed comparison.

Other Instantiations. We mainly focus on the DDH-based instantiation in this work. Nevertheless, our generic framework with $O(1)$ tracing can also be instantiated under other assumptions. For instance, as shown in [FLL+21], one may employ the LWE-based PRF scheme from [BPR12] together with the ZKPoK scheme described in [FLL+21], which in turn relies on the Stern protocol [LLNW17] and the accumulator construction [LLNW16] derived via the Unruh transform [Unr15]. A detailed treatment of these aspects, including the choice of parameters, is beyond the scope of this paper, and we leave the concrete realization of $O(1)$ tracing TRS from lattices as future work.

1.4 Roadmap

This paper is organized as follows. In Sec. 2 we present preliminaries and cryptographic tools used in this paper. In Sec. 3 we introduce TRS and present the (extended) security notions. In Sec. 4 we give the generic framework of TRS with $O(1)$ tracing. A concrete log-size scheme based on the DDH assumption is given in Sec. 5.

We defer additional supporting material to the appendix, including hardness assumptions in Appendix A and detailed proofs in Appendixes B and C. In Appendixes D and E we present several flaws in existing TRS schemes [WLB+24, WLB+23, QW22].

2 Preliminaries

Let $\lambda \in \mathbb{N}$ denote the security parameter. For $n \in \mathbb{Z}^+$, define $[n] = \{1, 2, ..., n\}$. Denote by $x := y$ the operation of assigning y to x. Denote by $x \xleftarrow{\$} S$ the operation of sampling x uniformly at random from a set S. For a distribution $\mathcal{D}$, denote by $x \leftarrow \mathcal{D}$ the operation of sampling x according to $\mathcal{D}$. For an algorithm $\mathcal{A}$, denote by $y \leftarrow \mathcal{A}(x; r)$ or $y := \mathcal{A}(x; r)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning the output to y. If the randomness is uniformly sampled, we simply denote as $y \leftarrow \mathcal{A}(x)$. For

deterministic algorithms $\mathcal{A}$, we also write as $y := \mathcal{A}(x)$ or $y := \mathcal{A}(x; r)$. By $[\![A]\!]$ we denote the bit that is 1 if the evaluation of the boolean statement A is true and 0 otherwise.

For a ring of users R, we use PK_R to denote the public key list of all users in R. "PPT" is short for probabilistic polynomial-time. A function $f(\lambda)$ is negligible, if for any polynomial $p(\lambda)$, there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$, we have $f(\lambda) < \frac{1}{p(\lambda)}$.

We defer the DLog and DDH assumptions to Appendix A.

2.1 Pseudorandom Functions

Definition 1 (Pseudorandom Functions). *A family of Pseudurandom Functions (PRF) is a family of functions $\{F_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ satisfying the following properties.*

- *Given $k \in \mathcal{K}$ and $x \in \mathcal{X}$, the computation of $F_k(x)$ is efficient.*
- *For any PPT adversary $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{prf}_{F,\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}^{F_k(\cdot)} = 1] - \Pr[\mathcal{A}^{R(\cdot)} = 1]| \leq negl(\lambda),$$

where $k \leftarrow \mathcal{K}$, and R is a random function from $\mathcal{X}$ to $\mathcal{Y}$.

We define the following uniqueness, which is tailored from [FLL+21] and adapted to our generic TRS construction.

Definition 2 (Uniqueness of PRF). *We say a PRF scheme has uniqueness, if for any PPT adversary $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{uniq}_{F,\mathcal{A}}(\lambda) := \Pr[(x, k_1, k_2) \leftarrow \mathcal{A} \ : \ k_1 \neq k_2 \wedge F_{k_1}(x) = F_{k_2}(x)] \leq negl(\lambda).$$

The following two (weaker) variants follow immediately from uniqueness.

$$\mathsf{Adv}^{uniq\text{-}v1}_{F,\mathcal{A}}(\lambda) := \Pr[k_1 \leftarrow \mathcal{K}; (x, k_2) \leftarrow \mathcal{A}^{F_{k_1}(\cdot)} \ : \ k_1 \neq k_2 \wedge F_{k_1}(x) = F_{k_2}(x)] \leq negl(\lambda).$$

$$\mathsf{Adv}^{uniq\text{-}v2}_{F,\mathcal{A}}(\lambda) := \Pr[x \leftarrow \mathcal{X} : (k_1, k_2) \leftarrow \mathcal{A} \ : \ k_1 \neq k_2 \wedge F_{k_1}(x) = F_{k_2}(x)] \leq negl(\lambda).$$

2.2 Zero-Knowledge Proof of Knowledge Arguments

Let R be a binary relation, and let $\mathcal{L}_R : \{x \mid \exists w \ s.t. \ (x, w) \in R\}$ be the language corresponding to R.

Definition 3 (ZKPoK). *A Zero-Knowledge Proof of Knowledge Arguments (ZKPoK) scheme* ZKPoK *for language $\mathcal{L}_R$ consists of the following three algorithms.*

- $\mathsf{crs} \xleftarrow{\$} \mathsf{GenCRS}(1^\lambda)$*: The CRS generation algorithm takes the security parameter 1^λ as input, and outputs a common reference string* crs*.*
- $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)$*: The proving algorithm takes* crs*, instance x and the corresponding witness w as input, and outputs a proof π.*
- $b \leftarrow \mathsf{Ver}(\mathsf{crs}, x, \pi)$*: The verification algorithm takes* crs*, x, and π as input, and outputs a bit indicating the validity of π w.r.t. x.*

Completeness. *For any* $\mathsf{crs} \xleftarrow{\$} \mathsf{GenCRS}(1^\lambda)$*, $(x, w) \in \mathcal{L}_R$, and $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$, it holds that* $\mathsf{Ver}(\mathsf{crs}, x, \pi) = 1$.

Definition 4 (Zero-knowledge of ZKPoK). *Consider the experiment* $\mathsf{Exp}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{A}}(\lambda)$ *defined in Fig. 2. We say a ZKPoK scheme* ZKPoK *for $\mathcal{L}_R$ has zero knowledge, if there is a simulator* Sim *such that for any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{A}}(\lambda) := |2 \cdot \Pr[\mathsf{Exp}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{A}}(\lambda) = 1] - 1| \leq negl(\lambda).$$

Definition 5 (Strong Simulation Extractability of ZKPoK). *Consider the experiment* $\mathsf{Exp}^{sse}_{\mathsf{ZKPoK},\mathsf{Sim},\mathsf{Ext},\mathcal{A}}(\lambda)$ *defined in Fig. 2. We say a ZKPoK scheme* ZKPoK *for $\mathcal{L}_R$ has strong simulation extractability, if there is a simulator* Sim *and an extractor* Ext *such that, for any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{sse}_{\mathsf{ZKPoK},\mathsf{Sim},\mathsf{Ext},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{sse}_{\mathsf{ZKPoK},\mathsf{Sim},\mathsf{Ext},\mathcal{A}}(\lambda) = 1]| \leq negl(\lambda).$$

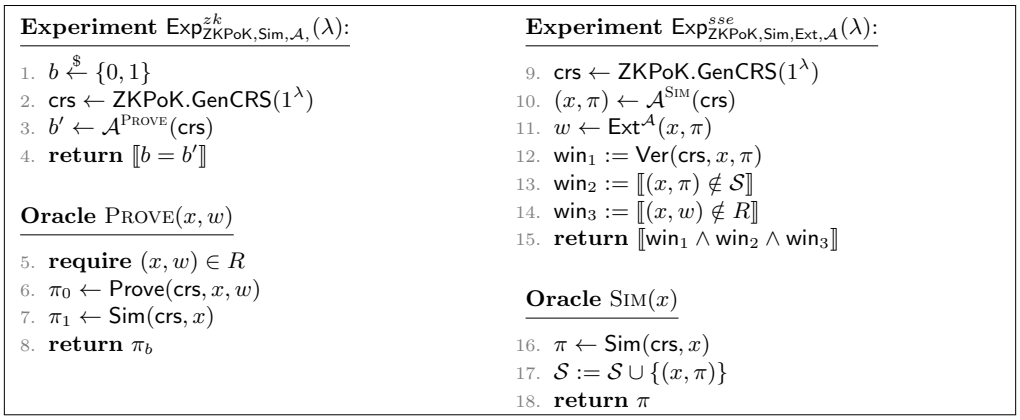

Fig. 2. Security experiments for ZKPoK scheme ZKPoK. Here both the simulator Sim and the extractor Ext may reprogram the random oracle(s) during the execution.

3 Traceable Ring Signatures

In this section we revisit the syntax and security notions of TRS.

Definition 6 (Traceable Ring Signatures). *An (event-orbited) traceable ring signature (TRS) scheme* $\mathsf{TRS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Trace})$ *consists of the following five algorithms.*

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes the security parameter* 1^λ *as input, and outputs a public parameter* pp*.*
- $(pk, sk) \leftarrow \mathsf{Gen}(pp)$*: The key generation algorithm takes* pp *as input, and outputs a public key* pk *and a secret key* sk*.*
- $\sigma \leftarrow \mathsf{Sign}(PK_R, sk_\delta, \mathsf{e}, \mathsf{m})$*: The signing algorithm takes as input a set of public keys* PK_R*, the secret key* sk_δ *of user* $\delta \in R$*, an event label* e *and a message* m*, and outputs a signature* σ*.*
- $b \leftarrow \mathsf{Ver}(PK_R, \mathsf{e}, \mathsf{m}, \sigma)$*: The verification algorithm* Ver *takes as input* PK_R*,* e*,* m *and* σ*, and outputs a bit indicating the validity of* σ*.*
  We say a signature σ *(w.r.t. ring* R*, event* e*, message* m*) is valid (resp. invalid), if* $\mathsf{Ver}(PK_R, \mathsf{e}, \mathsf{m}, \sigma) = 1$ *(resp.* $\mathsf{Ver}(PK_R, \mathsf{e}, \mathsf{m}, \sigma) = 0$*).*
- $\mathsf{res} \leftarrow \mathsf{Trace}(\mathsf{e}, PK_R, \mathsf{m}, \sigma, PK_{R'}, \mathsf{m}', \sigma')$*: The trace algorithm that takes as input an event label* e *and two signatures* σ *and* σ' *that correspond to* (PK_R, m) *and* $(PK_{R'}, \mathsf{m}')$*, respectively, and outputs a result* $\mathsf{res} \in \{\mathsf{indep}, \mathsf{linked}\} \cup \{\mathsf{linked} \times \mathcal{PK}\}$*, where* $\mathcal{PK}$ *denotes the space of public keys.*

Correctness. *For any rings* R, R'*, any* $\delta \in R$ *and* $\delta' \in R'$*, any event* e *and messages* m, m'*:*

1. *If* $pp \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(sk_i, pk_i) \leftarrow \mathsf{Gen}(pp)$ *for* $i \in R$*,* $\sigma \leftarrow \mathsf{Sign}(PK_R, sk_\delta, \mathsf{e}, \mathsf{m})$*, then it holds with overwhelming probability that* $\mathsf{Ver}(PK_R, \mathsf{e}, \mathsf{m}, \sigma) = 1$*.*
2. *If* $pp \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(sk_i, pk_i) \leftarrow \mathsf{Gen}(pp)$ *for* $i \in R \cup R'$*,* $\sigma \leftarrow \mathsf{Sign}(PK_R, sk_\delta, \mathsf{e}, \mathsf{m})$ *and* $\sigma' \leftarrow \mathsf{Sign}(PK_{R'}, sk_{\delta'}, \mathsf{e}, \mathsf{m}')$*, then it holds with overwhelming probability that*

$$\mathsf{Trace}(\mathsf{e}, PK_R, \mathsf{m}, \sigma, PK_{R'}, \mathsf{m}', \sigma') = \begin{cases} \mathsf{indep} & \text{if } \delta \neq \delta', \\ \mathsf{linked} & \text{else if } \mathsf{m} = \mathsf{m}', \\ (\mathsf{linked}, pk_\delta) & \text{otherwise.} \end{cases}$$

Remark 1 (Event Labels). Most existing TRS works adopt the definition in [FS07], where Trace is defined on a tag $L = (\mathsf{issue}, PK_R)$ with issue a signing issue. Consequently, $\mathsf{Trace}(\sigma, \sigma')$ functions only when σ and σ' correspond to the same issue and ring. However, this limits the applications of TRS where PK_R differs in each signing call. For example, in e-voting, the registration list of PK_R may dynamically update until the

end of the voting, making it unrealistic to assume that all voters cast their ballots simultaneously and select the same ring.

In this work we adopt a more general syntax, defining Trace associated to an event label $\mathsf{e} \in \{0,1\}^*$ alone, as the work already done in Linkable Ring Signatures (LRS) [TWC+04, LASZ13, CLXZ25]. This yields traceability across signatures that share an event but are associated with *different* rings. By defining $\mathsf{e} := (\mathsf{issue}, PK_R)$, it covers the definition of [FS07].

In this paper we regard $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') = (\mathsf{linked}, pk)$ as a special case of linked, and sometimes simply denote the result by pk. For simplicity, we also write $\mathsf{Trace}(\mathsf{e}, PK_R, \mathsf{m}, \sigma, PK_{R'}, \mathsf{m}', \sigma')$ as $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma')$ whenever the remaining information is either irrelevant or can be deduced from the context.

We extend the syntax of TRS so that the trace algorithm works for any two signatures w.r.t. different rings and messages, as long as they are related to the same event label e[2]—that explains why it is called "event-orbited". By defining e to be the ring or the concatenation of the ring and an issue label, it covers the definition in [WLB+24] and the definition in [FS07, BM19, FLWL20, FLL+21, KSD+24].

Fact 1 *Assume whether* $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') = \mathsf{indep}$ *is determined by comparing a designated component of* σ *and* σ'*, then* $\mathsf{Trace}(\mathsf{e}, \cdot, \cdot) = \mathsf{linked}$*, including the case that a public key is traced, defines an equivalence relation for signatures w.r.t. an event label* e*. Moreover, it has the following transitivity:*

For any e *and* $\sigma, \sigma', \sigma''$*, if* $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') = \mathsf{linked}$ *and* $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma'') = \mathsf{linked}$*, then* $\mathsf{Trace}(\mathsf{e}, \sigma', \sigma'') = \mathsf{linked}$*.*

As noted in prior work, $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') = \mathsf{linked}$ does not necessarily mean that σ and σ' were generated by the same signer, since anyone can copy an existing signature verbatim. Such "duplication attacks", however, are easily filtered out in practice.

Moreover, if the TRS scheme is malleable (e.g., it achieves unforgeability but not strong unforgeability), then given σ, it is feasible to generate another valid σ' w.r.t. the same ring R and message m such that $\mathsf{Trace}(\sigma, \sigma') = \mathsf{linked}$. To fence this in applications like voting systems, m is usually required to contain a unique (and random) serial number.

3.1 Security Notions of TRS

We require extended linkability, extended exculpability, and anonymity for TRS.

- **Extended Linkability**. For a fixed event label, an adversary corrupting ℓ users cannot generate $\ell + 1$ signatures such that either (1) all signatures are pairwise independent; or (2) all except two signatures are pairwise independent, and the two signatures on different messages fail to trace to one corrupted public key.
  Note that most prior work [FS07, FLL+21, WLB+24, KSD+24] considered a weaker notion of linkability, referred to as variant linkability in Fig. 4, where an adversary cannot generate $\ell+1$ pairwise independent signatures for a ring of size ℓ. Since the number of corrupted users ranges from 0 to the ring size, the variant linkability is covered by condition (1) of our new definition. In Sec. 3.2, we formally prove that the weaker notion together with extended exculpability imply condition (1) in our framework.
- **Extended Exculpability**. For a fixed event label, an adversary can neither (1) output two signatures (of which at least one is adversarial) that trace to an honest user; nor (2) forge a signature that is linked to one existing signature of an honest user.
  Note that most prior works [FS07, FLL+21, WLB+24, KSD+24] considered only condition (1) but not condition (2). The latter captures the attack in which the adversary forges $(R', \mathsf{e}, \mathsf{m}', \sigma')$ after observing $(R, \mathsf{e}, \mathsf{m}, \sigma)$ from an honest user with pk. Condition (1) guarantees only that $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') \neq pk$. However, it may still hold that $\mathsf{Trace}(\mathsf{e}, \sigma, \sigma') = \mathsf{linked}$ (for example, $\mathsf{m}' = \mathsf{m}$ but $R' \neq R$). In applications like e-voting, this attack would cause both σ and σ' to be deemed invalid, thereby disenfranchising honest voters.

[2] The event label e here is also referred as a tag, a prefix [BEHM22, HC24], a scope [HKSS22], or a key-image [KWT25].

– **Anonymity**. The traceable ring signature does not reveal the identity of the signer.

We formally define these security notions as follows.

Definition 7 ((Extended) Linkability). *Consider the experiments* $\mathsf{Exp}^{(ex\text{-})link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ *defined in Fig. 3. We say a TRS scheme* TRS *has (extended) linkability, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{(ex\text{-})link}_{\mathsf{TRS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{(ex\text{-})link}_{\mathsf{TRS},\mathcal{A}}(\lambda) = 1] \leq negl(\lambda).$$

Definition 8 ((Extended) Exculpability). *Consider the experiments* $\mathsf{Exp}^{(ex)\text{-}excu}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ *defined in Fig. 3. We say a TRS scheme* TRS *has (extended) exculpability, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{(ex\text{-})excu}_{\mathsf{TRS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{(ex\text{-})excu}_{\mathsf{TRS},\mathcal{A}}(\lambda) = 1] \leq negl(\lambda).$$

Definition 9 (Anonymity). *Consider the experiment* $\mathsf{Exp}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ *defined in Fig. 3. We say a TRS scheme* TRS *has anonymity, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda) := |2 \cdot \Pr[\mathsf{Exp}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda) = 1] - 1| \leq negl(\lambda).$$

3.2 Unforgeability and A Variant Definition of Linkability

In this section we show that (extended) linkability and extended exculpability together imply unforgeability. This implication was first claimed by Fujisaki and Suzuki [FS07], but their proof is flawed, as discussed in Sec. 1.3. We repair it using our new notion of extended exculpability.

We first provide the formal definition of unforgeability.

Definition 10 (Unforgeability). *Consider the experiment* $\mathsf{Exp}^{unforg}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ *defined in Fig. 4. We say a TRS scheme* TRS *has unforgeability, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{unforg}_{\mathsf{TRS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{unforg}_{\mathsf{TRS},\mathcal{A}}(\lambda) = 1] \leq negl(\lambda).$$

Theorem 1. *If a TRS scheme* TRS *has superpolynomial message space, linkability and extended exculpability, then it also has unforgeability. More precisely, we have*

$$\mathsf{Adv}^{unforg}_{\mathsf{TRS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{ex\text{-}excu}_{\mathsf{TRS},\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{link}_{\mathsf{TRS},\mathcal{A}_2}(\lambda).$$

Proof (of Theorem 1). Suppose that an adversary $\mathcal{A}$ breaks the unforgeability of TRS. We show that there exists a reduction algorithm $\mathcal{A}'$ that either breaks the linkability, or the extended exculpability of TRS.

The reduction algorithm $\mathcal{A}'$ has access to oracles NewUser, Corr, NewCorr, Sign from its own challenger, and it simulates the unforgeability experiment for $\mathcal{A}$ (note that the (extended) linkability and extended exculpability experiments differ only in the final outputs and winning conditions). Let $(R^*, \mathsf{e}^*, \mathsf{m}^*, \sigma^*)$ be $\mathcal{A}$'s final output during the unforgeability experiment, and $R^* = \{n_1, ..., n_\ell\}$. Then $\mathcal{A}'$ samples ℓ messages $(\mathsf{m}_1, ..., \mathsf{m}_\ell)$ that differ from m^* and all messages appear in the simulation of Sign, and queries $\textsc{Sign}(n_i, \{n_i\}, \mathsf{e}^*, \mathsf{m}_i)$ for $i \in [\ell]$ to obtain the corresponding signatures $(\sigma_1, ..., \sigma_\ell)$.[3] Now we analyze in two cases.

– **Case 1:** There exists $i^*, j^* \in [\ell]$ such that $\mathsf{Trace}(\mathsf{e}^*, \sigma_{i^*}, \sigma^*) = \mathsf{linked}$, or $\mathsf{Trace}(\mathsf{e}^*, \sigma_{j^*}, \sigma^*) = pk_{i^*}$. By the correctness we have $\mathsf{Ver}(PK_{i^*}, \mathsf{e}^*, \mathsf{m}_{i^*}, \sigma_{i^*}) = 1$. Together with the fact that $\mathsf{Ver}(PK_{R^*}, \mathsf{e}^*, \mathsf{m}^*, \sigma^*) = 1$ and all secret keys in R^* are uncorrupted, we know $(\mathsf{win}_1 \wedge \mathsf{win}_2)$ is satisfied in the extended exculpability experiment. Meanwhile, $(\mathsf{win}_5 \wedge \mathsf{win}_6)$ or $(\mathsf{win}_3 \wedge \mathsf{win}_4)$ is satisfied since $\mathcal{A}'$ has never queried $\textsc{Sign}(\cdot, R^*, \mathsf{e}^*, \mathsf{m}^*)$, and $\mathsf{Trace}(\mathsf{e}^*, \sigma_{i^*}, \sigma^*) = \mathsf{linked}$ or $\mathsf{Trace}(\mathsf{e}^*, \sigma_{j^*}, \sigma^*) = pk_{i^*}$. Therefore, $\mathcal{A}'$ successfully forges a pair of linked signatures $(i^*, (R_{i^*}, \mathsf{e}^*, \mathsf{m}_{i^*}, \sigma_{i^*}), (R^*, \mathsf{e}^*, \mathsf{m}^*, \sigma^*))$ for $(\mathsf{win}_5 \wedge \mathsf{win}_6)$ or $(i^*, (R_{j^*}, \mathsf{e}^*, \mathsf{m}_{j^*}, \sigma_{j^*}), (R^*, \mathsf{e}^*, \mathsf{m}^*, \sigma^*))$ (for $\mathsf{win}_3 \wedge \mathsf{win}_4$), and hence breaks the extended exculpability.

[3] This step requires the message space to be superpolynomial, which, of course, is satisfied by almost all (traceable ring) signature schemes.

Experiments $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ and $\boxed{\mathsf{Exp}^{ex\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda)}$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ctr} := 0$; $\mathsf{win} := 0$; $\mathcal{S} := \varnothing$; $\mathcal{S}_{\mathsf{corr}} := \varnothing$; $\mathcal{S}_{\mathsf{cpk}} := \varnothing$
$(\mathsf{e}^*, \{(R_j, \mathsf{m}_j, \sigma_j)\}_{j\in[\ell+1]}\boxed{, \mu, \nu}) \leftarrow \mathcal{A}^{\textsc{NewUser},\textsc{Corr},\textsc{NewCorr},\textsc{Sign}}(pp)$
$\mathsf{win}_1 := [\![\forall j \in [\ell+1] : \mathsf{Ver}(PK_{R_j}, \mathsf{e}^*, \mathsf{m}_j, \sigma_j) = 1]\!]$
$\mathsf{win}_2 := [\![\forall j \in [\ell+1] : (\cdot, R_j, \mathsf{e}^*, \mathsf{m}_j, \cdot) \notin \mathcal{S}]\!]$
$\mathsf{win}_3 := [\![|\mathcal{S}_{\mathsf{corr}} \cap (\bigcup_{j\in[\ell+1]} R_j)| \le \ell]\!]$
$\mathsf{win}_4 := \left[\!\!\left[\begin{array}{l} \forall j,k \in [\ell+1], j \neq k : \\ \quad \mathsf{Trace}(\mathsf{e}^*, PK_{R_j}, \mathsf{m}_j, \sigma_j, PK_{R_k}, \mathsf{m}_k, \sigma_k) = \mathsf{indep} \end{array} \right]\!\!\right]$
// win_4: $\mathcal{A}$ gets $\ell+1$ pairwise indep signatures from at most ℓ corrupted users
$\mathsf{res} \leftarrow \mathsf{Trace}(\mathsf{e}^*, PK_{R_\mu}, \mathsf{m}_\mu, \sigma_\mu, PK_{R_\nu}, \mathsf{m}_\nu, \sigma_\nu)$
$\mathsf{win}_5 := \left[\!\!\left[\begin{array}{l} \forall j,k \in [\ell+1], j \neq k, (j,k) \neq (\mu,\nu) : \\ \quad \mathsf{Trace}(\mathsf{e}^*, PK_{R_j}, \mathsf{m}_j, \sigma_j, PK_{R_k}, \mathsf{m}_k, \sigma_k) = \mathsf{indep} \end{array} \right]\!\!\right]$
// win_5: all signatures except the μ-th and the ν-th are pairwise independent (weaker than win_4)
$\mathsf{win}_6 := [\![\mu, \nu \in [\ell+1] \wedge \mathsf{m}_\mu \neq \mathsf{m}_\nu]\!]$
$\mathsf{win}_7 := [\![\neg\left(\mathsf{res} = (\mathsf{linked}, pk) \wedge pk \in \mathcal{S}_{\mathsf{cpk}} \cap PK_{R_\mu} \cap PK_{R_\nu}\right)]\!]$
// $\mathsf{win}_5 \wedge \mathsf{win}_6 \wedge \mathsf{win}_7$: $\mathcal{A}$ forges two signatures σ_μ, σ_ν w.r.t. different messages that are linked but fail to trace
$\mathsf{win} := [\![(\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}_3) \wedge (\mathsf{win}_4\boxed{\vee(\mathsf{win}_5 \wedge \mathsf{win}_6 \wedge \mathsf{win}_7)})]\!]$
return win

Experiments $\mathsf{Exp}^{excu}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ and $\boxed{\mathsf{Exp}^{ex\text{-}excu}_{\mathsf{TRS},\mathcal{A}}(\lambda)}$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ctr} := 0$; $\mathsf{win} := 0$; $\mathcal{S} := \varnothing$; $\mathcal{S}_{\mathsf{corr}} := \varnothing$; $\mathcal{S}_{\mathsf{cpk}} := \varnothing$
$(i^*, (R, \mathsf{e}^*, \mathsf{m}, \sigma), (R', \mathsf{e}^*, \mathsf{m}', \sigma')) \leftarrow \mathcal{A}^{\textsc{NewUser},\textsc{Corr},\textsc{NewCorr},\textsc{Sign}}(pp)$
$\mathsf{res} \leftarrow \mathsf{Trace}(\mathsf{e}^*, PK_R, \mathsf{m}, \sigma, PK_{R'}, \mathsf{m}', \sigma')$
$\mathsf{win}_1 := [\![\mathsf{Ver}(PK_R, \mathsf{e}^*, \mathsf{m}, \sigma) = 1 \wedge \mathsf{Ver}(PK_{R'}, \mathsf{e}^*, \mathsf{m}', \sigma') = 1]\!]$
$\mathsf{win}_2 := [\![\mathsf{corr}[i^*] = 0]\!]$
$\mathsf{win}_3 := [\![(i^*, R', \mathsf{e}^*, \mathsf{m}', \cdot) \notin \mathcal{S}]\!]$
$\mathsf{win}_4 := [\![(pk, sk) \leftarrow \mathsf{key}[i^*] : \mathsf{res} = (\mathsf{linked}, pk)]\!]$
// $\mathsf{win}_3 \wedge \mathsf{win}_4$: $\mathcal{A}$ forges one signature that traced to an honest user
$\mathsf{win}_5 := [\![(i^*, R, \mathsf{e}^*, \mathsf{m}, \sigma) \in \mathcal{S} \wedge (i^*, R', \mathsf{e}^*, \mathsf{m}', \cdot) \notin \mathcal{S}]\!]$
$\mathsf{win}_6 := [\![\mathsf{res} = \mathsf{linked}]\!]$
// $\mathsf{win}_5 \wedge \mathsf{win}_6$: $\mathcal{A}$ forges one signature that is linked to an existing one
$\mathsf{win} := [\![(\mathsf{win}_1 \wedge \mathsf{win}_2) \wedge ((\mathsf{win}_3 \wedge \mathsf{win}_4)\boxed{\vee(\mathsf{win}_5 \wedge \mathsf{win}_6)})]\!]$
return win

Experiment $\mathsf{Exp}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda)$:

$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $b \leftarrow \{1,2\}$
$\mathsf{ctr} := 0$; $\mathsf{win} := 0$; $\mathcal{S} := \varnothing$; $\mathcal{S}_{\mathsf{corr}} := \varnothing$; $\mathcal{S}_{\mathsf{cpk}} := \varnothing$; $\mathcal{S}_{\mathsf{chall}} := \varnothing$
NewUser(); NewUser()
// initiate the 1-st and the 2-nd users as the challenge honest users
$b' \leftarrow \mathcal{A}^{\textsc{NewUser},\textsc{Corr},\textsc{NewCorr},\textsc{Sign},\textsc{Chall}}(pp)$
require $(\mathsf{corr}[1] = \mathsf{corr}[2] = 0)$
require $\forall(\cdot,\cdot,\mathsf{e},\cdot,\cdot) \in \mathcal{S}_{\mathsf{chall}} : (1,\cdot,\mathsf{e},\cdot,\cdot) \notin \mathcal{S} \wedge (2,\cdot,\mathsf{e},\cdot,\cdot) \notin \mathcal{S}$
// $\mathcal{A}$ cannot query $\textsc{Sign}(\{1,2\},)$ with the same event e in Chall()
return $[\![b = b']\!]$

Oracle NewUser()

$\mathsf{ctr}{+}{+}$
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathsf{key}[\mathsf{ctr}] := (pk, sk)$
$\mathsf{corr}[\mathsf{ctr}] := 0$
return pk

Oracle Corr(i)

require $(i \le \mathsf{ctr}) \wedge (\mathsf{corr}[i] = 0)$
$\mathsf{corr}[i] := 1$
$(pk, sk) \leftarrow \mathsf{key}[i]$
$\mathcal{S}_{\mathsf{corr}} := \mathcal{S}_{\mathsf{corr}} \cup \{i\}$
$\mathcal{S}_{\mathsf{cpk}} := \mathcal{S}_{\mathsf{cpk}} \cup \{pk\}$
return sk

Oracle NewCorr(pk)

$\mathsf{ctr}{+}{+}$
$\mathsf{key}[\mathsf{ctr}] := (pk, \perp)$
$\mathsf{corr}[\mathsf{ctr}] := 1$
$\mathcal{S}_{\mathsf{corr}} := \mathcal{S}_{\mathsf{corr}} \cup \{\mathsf{ctr}\}$
$\mathcal{S}_{\mathsf{cpk}} := \mathcal{S}_{\mathsf{cpk}} \cup \{pk\}$
return

Oracle Sign($i, R, \mathsf{e}, \mathsf{m}$)

require $(R \subseteq [\mathsf{ctr}]) \wedge (i \in R)$
require $(\mathsf{corr}[i] = 0)$
$(pk, sk) \leftarrow \mathsf{key}[i]$
$\sigma \leftarrow \mathsf{Sign}(PK_R, sk, \mathsf{e}, \mathsf{m})$
$\mathcal{S} := \mathcal{S} \cup (i, R, \mathsf{e}, \mathsf{m}, \sigma)$
return σ

Oracle Chall($R, \mathsf{e}, \mathsf{m}$)

// challenge query w.r.t. honest users (the 1-st and the 2-nd)
require $\{1,2\} \subseteq R \subseteq [\mathsf{ctr}]$
require $(\mathsf{corr}[1] = \mathsf{corr}[2] = 0)$
require $\nexists(\cdot,\cdot,\mathsf{e},\mathsf{m}' \neq \mathsf{m},\cdot) \in \mathcal{S}_{\mathsf{chall}}$
// $\mathcal{A}$ cannot query Chall twice with the same e but different m, m'
$(pk, sk) \leftarrow \mathsf{key}[b]$
$\sigma \leftarrow \mathsf{Sign}(PK_R, sk, \mathsf{e}, \mathsf{m})$
$\mathcal{S}_{\mathsf{chall}} := \mathcal{S}_{\mathsf{chall}} \cup \{(b, R, \mathsf{e}, \mathsf{m}, \sigma)\}$
return σ

Fig. 3. Security experiments of (extended) linkability, (extended) exculpability, and anonymity for TRS. All sets are initialized to be empty, and all integer variables are initialized to be 0. The extended parts of linkability and exculpability are highlighted in the box.

Experiment $\mathsf{Exp}^{unforg}_{\mathsf{TRS},\mathcal{A}}(\lambda)$:

1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$
2. $\mathsf{ctr} := 0$; $\mathsf{win} := 0$; $\mathcal{S} := \varnothing$; $\mathcal{S}_{\mathsf{corr}} := \varnothing$; $\mathcal{S}_{\mathsf{cpk}} := \varnothing$
3. $(\mathsf{e}^*, R^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{NewUser},\textsc{Corr},\textsc{NewCorr},\textsc{Sign}}(pp)$
4. $\mathsf{win}_1 := [\![\mathsf{Ver}(PK_{R^*}, \mathsf{e}^*, \mathsf{m}^*, \sigma^*) = 1]\!]$
5. $\mathsf{win}_2 := [\![\forall i \in R^* : \mathsf{corr}[i] = 0]\!]$
6. $\mathsf{win}_3 := [\![(\cdot, R^*, \mathsf{e}^*, \mathsf{m}^*, \cdot) \notin \mathcal{S}]\!]$
7. $\mathsf{win} := [\![\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}_3]\!]$
8. **return** win

Experiment $\mathsf{Exp}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$:
// def. of variant linkability in [FS07]

9. $pp \leftarrow \mathsf{Setup}(1^\lambda)$
10. $\mathsf{ctr} := 0$; $\mathsf{win} := 0$; $\mathcal{S} := \varnothing$; $\mathcal{S}_{\mathsf{corr}} := \varnothing$; $\mathcal{S}_{\mathsf{cpk}} := \varnothing$
11. $(\mathsf{e}^*, \{(R_j, \mathsf{m}_j, \sigma_j)\}_{j\in[\ell+1]}) \leftarrow \mathcal{A}^{\textsc{NewUser},\textsc{Corr},\textsc{NewCorr},\textsc{Sign}}(pp)$
12. $\mathsf{win}_1 := [\![\forall j \in [\ell+1] : \mathsf{Ver}(PK_{R_j}, \mathsf{e}^*, \mathsf{m}_j, \sigma_j) = 1]\!]$
13. $\mathsf{win}_2 := [\![\forall j \in [\ell+1] : (\cdot, R_j, \mathsf{e}^*, \mathsf{m}_j, \cdot) \notin \mathcal{S}]\!]$
14. $\mathsf{win}'_3 := [\![|\bigcup_{j\in[\ell+1]} R_j| \le \ell]\!]$
15. // in extended linkability def. $\mathsf{win}_3 := [\![|\mathcal{S}_{\mathsf{corr}} \cap (\bigcup_{j\in[\ell+1]} R_j)| \le \ell]\!]$
16. $\mathsf{win}_4 := \left[\!\!\left[\begin{array}{l} \forall j,k \in [\ell+1], j \neq k : \\ \quad \mathsf{Trace}(\mathsf{e}^*, PK_{R_j}, \mathsf{m}_j, \sigma_j, PK_{R_k}, \mathsf{m}_k, \sigma_k) = \mathsf{indep} \end{array} \right]\!\!\right]$
17. // win_4: $\mathcal{A}$ gets $\ell+1$ pairwise indep signatures associate with rings of size at most ℓ
18. $\mathsf{win} := [\![\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}'_3 \wedge \mathsf{win}_4]\!]$
19. **return** win

Oracle $\textsc{NewUser}()$

20. $++\,\mathsf{ctr}$
21. $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
22. $\mathsf{key}[\mathsf{ctr}] := (pk, sk)$
23. $\mathsf{corr}[\mathsf{ctr}] := 0$
24. **return** pk

Oracle $\textsc{Corr}(i)$

25. **require** $(i \le \mathsf{ctr}) \wedge (\mathsf{corr}[i] = 0)$
26. $\mathsf{corr}[i] := 1$
27. $(pk, sk) \leftarrow \mathsf{key}[i]$
28. $\mathcal{S}_{\mathsf{corr}} := \mathcal{S}_{\mathsf{corr}} \cup \{i\}$
29. $\mathcal{S}_{\mathsf{cpk}} := \mathcal{S}_{\mathsf{cpk}} \cup \{pk\}$
30. **return** sk

Oracle $\textsc{NewCorr}(pk)$

31. $++\,\mathsf{ctr}$
32. $\mathsf{key}[\mathsf{ctr}] := (pk, \perp)$
33. $\mathsf{corr}[\mathsf{ctr}] := 1$
34. $\mathcal{S}_{\mathsf{corr}} := \mathcal{S}_{\mathsf{corr}} \cup \{\mathsf{ctr}\}$
35. $\mathcal{S}_{\mathsf{cpk}} := \mathcal{S}_{\mathsf{cpk}} \cup \{pk\}$
36. **return**

Oracle $\textsc{Sign}(i, R, \mathsf{e}, \mathsf{m})$

37. **require** $(R \subseteq [\mathsf{ctr}]) \wedge (i \in R)$
38. **require** $(\mathsf{corr}[i] = 0)$
39. $(pk, sk) \leftarrow \mathsf{key}[i]$
40. $\sigma \leftarrow \mathsf{Sign}(PK_R, sk, \mathsf{e}, \mathsf{m})$
41. $\mathcal{S} := \mathcal{S} \cup (i, R, \mathsf{e}, \mathsf{m}, \sigma)$
42. **return** σ

Fig. 4. Security experiments of unforgeability, and linkability defined in [FS07]. Note that oracles NewUser, Corr, NewCorr, Sign are the same as in Fig. 3. All sets are initialized to be empty, and all integer variants are initialized to be 0. The different (and weakened) part of linkability is highlighted in gray.

– **Case 2:** For all $i \in [\ell]$ it holds that $\mathsf{Trace}(\mathsf{e}^*, \sigma_i, \sigma^*) = \mathsf{indep}$.
In this case, $\mathcal{A}'$ queries $\textsc{Corr}(i)$ for all $i \in [\ell]$ to obtain the secret keys $\{sk_{n_i}\}_{i\in[\ell]}$. Then, it samples another ℓ fresh messages $(\mathsf{m}'_1, ..., \mathsf{m}'_\ell)$ that differ from m^*, $(\mathsf{m}_1, ..., \mathsf{m}_\ell)$ and all messages appear in the simulation of Sign, and invokes $\mathsf{Sign}(\{pk_{n_i}\}, sk_{n_i}, \mathsf{e}^*, \mathsf{m}'_i)$ to obtain σ'_i. Finally $\mathcal{A}'$ outputs $(\mathsf{e}^*, \{(\{n_i\}, \mathsf{m}'_i, \sigma'_i)\}_{i\in[\ell]}, (R^*, \mathsf{m}^*, \sigma^*))$ as its final attack against the linkability.
By the correctness, all $\ell+1$ signatures are valid, i.e., win_1 in $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}'}(\lambda)$ is satisfied. Meanwhile, it is straightforward to see that all signatures are not related to Sign and $|\mathcal{S}_{\mathsf{corr}} \cap (\bigcup R_i \cup R^*)| = \ell$, indicating that win_2 and win_3 are satisfied. Nevertheless, for all $i \in [\ell]$, we have $\mathsf{Trace}(\mathsf{e}^*, \sigma_i, \sigma'_i) = \mathsf{linked}$ (by correctness) and $\mathsf{Trace}(\mathsf{e}^*, \sigma_i, \sigma^*) = \mathsf{indep}$. Therefore, by Fact 1 we know $\mathsf{Trace}(\mathsf{e}^*, \sigma'_i, \sigma^*) = \mathsf{indep}$ for all $i \in [\ell]$. As a result, win_4 is satisfied and $\mathcal{A}'$ successfully breaks the linkability, concluding the proof.

□

Now we recall the variant definition of linkability in [FS07, FLL+21, WLB+24, KSD+24] and compare it with our notions. Informally, the variant definition requires that an adversary cannot make $\ell+1$ pairwise independent signatures for a ring of size at most ℓ. Note that in [FS07, FLL+21, WLB+24, KSD+24], Trace is defined w.r.t. a tag and a ring, and in the security experiment all $\ell+1$ forgeries outputted by the adversary share the same tag and ring R. Therefore, the adversary wins as long as all $\ell+1$ forgeries are pairwise

independent and $|R| \leq \ell$. In this work we take a more general syntax of TRS by defining Trace w.r.t. event label e. Therefore, in Def. 11 we allow the $\ell+1$ signatures to be corresponding to different rings R_i, and the winning condition $|R| \leq \ell$ is replaced with the condition $|\bigcup R_i| \leq \ell$.

Definition 11 (Variant Linkability [FS07]). *Consider the experiments* $\mathsf{Exp}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ *defined in Fig. 4. We say a TRS scheme* TRS *has variant linkability, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda) = 1] \leq negl(\lambda).$$

Theorem 2. *If a TRS scheme* TRS *has linkability (Def. 7 and Fig. 3), then it has variant linkability (Def. 11 and Fig. 4). On the other hand, if a TRS scheme* TRS *has variant linkability and extended exculpability, then it also has linkability. More precisely, we have*

$$\mathsf{Adv}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{ex\text{-}excu}_{\mathsf{TRS},\mathcal{A}_1}(\lambda) + 2 \cdot \mathsf{Adv}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}_2}(\lambda).$$

The proof is deferred to Appendix B.

4 Generic Construction of TRS with $O(1)$ Tracing

In this section, we present our new generic construction of TRS. Compared to the scheme by Feng et al. [FLL+21], which follows the design by Fujisaki and Suzuki [FS07], our construction requires weaker security properties of PRF and achieves better efficiency, for example, $O(1)$ tracing.

Denote $\mathcal{PK}$ the public key space and $\mathcal{M} \subset \mathbb{Z}$ the message space. We assume $\mathcal{PK}$ is a module over a commutative ring, and that every non-zero element of $\mathsf{m} - \mathsf{m}'$ is invertible. We consider the following language for ZKPoK:

$$\begin{aligned}\mathcal{L}_{\mathsf{TRS}} := \{ins = (PK_R, \mathsf{e}, \mathsf{m}, \alpha, pid_1, pid_2) \mid \exists wit = (\delta, sk_\delta)\ s.t.\ \delta \in R \ \wedge\ pk_\delta = F_{sk_\delta}(\alpha) \in PK_R \\ \wedge\ pid_1 = F_{sk_\delta}(\mathsf{e}||1) \ \wedge\ pid_2 = F_{sk_\delta}(\mathsf{e}||2) + (\mathsf{m} \cdot pk_\delta)\}.\end{aligned}$$

We assume the domains of α, $(\mathsf{e}||1)$, and $(\mathsf{e}||2)$ are disjoint. Our generic construction is shown in Fig. 5.

$\underline{\mathsf{Setup}(1^\lambda)}$

1. $\alpha \leftarrow \mathcal{X}$; $\mathsf{crs} \leftarrow \mathsf{ZKPoK.GenCRS}(1^\lambda)$
2. **return** $pp := (\alpha, \mathsf{crs})$

$\underline{\mathsf{Gen}(pp)}$

3. $sk \leftarrow \mathsf{PRF.Gen}(1^\lambda)$
4. **return** $(pk := F_{sk}(\alpha), sk)$

$\underline{\mathsf{Sign}(PK_R, sk_\delta, \mathsf{e}, \mathsf{m})}$

5. $pid_1 := F_{sk_\delta}(\mathsf{e}||1)$
6. $pid_2 := F_{sk_\delta}(\mathsf{e}||2) + (\mathsf{m} \cdot pk_\delta)$
7. $ins := (PK_R, \mathsf{e}, \mathsf{m}, \alpha, pid_1, pid_2)$
8. $wit := (\delta, sk_\delta)$
9. $\pi \leftarrow \mathsf{ZKPoK.Prove}(\mathsf{crs}, ins, wit)$
10. **return** $\sigma := (pid_1, pid_2, \pi)$

$\underline{\mathsf{Ver}(PK_R, \mathsf{e}, \mathsf{m}, \sigma)}$

11. $\sigma = (pid_1, pid_2, \pi)$
12. $ins := (PK_R, \mathsf{e}, \mathsf{m}, \alpha, pid_1, pid_2)$
13. **return** $\mathsf{ZKPoK.Ver}(\mathsf{crs}, ins, \pi)$

$\underline{\mathsf{Trace}(\mathsf{e}, PK_R, \mathsf{m}, \sigma, PK_{R'}, \mathsf{m}', \sigma')}$

14. $\sigma = (pid_1, pid_2, ...)$; $\sigma' = (pid'_1, pid'_2, ...)$
15. **if** $pid_1 \neq pid'_1$: **return** indep
16. **else if** $(pid_1, pid_2) = (pid'_1, pid'_2)$:
17. **return** linked
18. **else if** $\mathsf{m} \neq \mathsf{m}'$:
19. **return** $(\mathsf{m} - \mathsf{m}')^{-1} \cdot (pid_2 - pid'_2)$
20. // This includes the case $pk_\delta = E$, the additive identity element of $\mathcal{PK}$
21. **else** : **return** indep

Fig. 5. Generic construction of TRS from PRF scheme F and ZKPoK scheme ZKPoK.

Correctness of Verification. This follows directly from the completeness of ZKPoK.

Correctness of Trace. For a fixed event label e, consider two signatures $\sigma = (pid_1, pid_2, \pi)$ and $\sigma' = (pid'_1, pid'_2, \pi')$ signed with sk_δ and $sk_{\delta'}$. By construction we know $pid_1 = F_{sk_\delta}(\mathsf{e}||1)$, $pid_2 = F_{sk_\delta}(\mathsf{e}||2) + (\mathsf{m} \cdot pk_\delta)$, $pid'_1 = F_{sk'_\delta}(\mathsf{e}||1)$, $pid'_2 = F_{sk'_\delta}(\mathsf{e}||2) + (\mathsf{m}' \cdot pk_{\delta'})$.

1. If $\delta \neq \delta'$, then Trace will output indep unless $F_{sk_\delta}(\mathsf{e}||1) = F_{sk_{\delta'}}(\mathsf{e}||1)$. Since sk_δ and $sk_{\delta'}$ are sampled independently, this happens with negligible probability due to the uniqueness of PRF.
2. If $\delta = \delta'$ and $\mathsf{m} = \mathsf{m}'$, then $(pid_1, pid_2) = (pid'_1, pid'_2)$ and Trace will return linked.
3. If $\delta = \delta'$ but $\mathsf{m} \neq \mathsf{m}'$, then Trace will return

$$(\mathsf{m} - \mathsf{m}')^{-1} \cdot (pid_2 - pid'_2) = (\mathsf{m} - \mathsf{m}')^{-1} \cdot (\mathsf{m} \cdot pk_\delta - \mathsf{m}' \cdot pk_\delta) = pk_\delta.$$

Theorem 3. *If the PRF scheme F is secure and has uniqueness, the ZKPoK scheme ZKPoK has zero-knowledge and strong simulation extractability, then the TRS scheme in Fig. 5 has extended linkability, extended exculpability, and anonymity. More precisely, we have*

$$\begin{aligned}
\mathsf{Adv}^{ex\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda) &\leq \mathsf{Adv}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{B}_1}(\lambda) + (\ell+1) \cdot \mathsf{Adv}^{sse}_{\mathsf{ZKPoK},\mathsf{Sim},\mathsf{Ext},\mathcal{B}_2}(\lambda) \\
&\quad + n \cdot \mathsf{Adv}^{prf}_{F,\mathcal{B}_3}(\lambda) + 2 \cdot \mathsf{Adv}^{uniq}_{F,\mathcal{B}_4}(\lambda), \\
\mathsf{Adv}^{ex\text{-}excu}_{\mathsf{TRS},\mathcal{A}}(\lambda) &\leq \mathsf{Adv}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{B}_1}(\lambda) + 2 \cdot \mathsf{Adv}^{sse}_{\mathsf{ZKPoK},\mathsf{Sim},\mathsf{Ext},\mathcal{B}_2}(\lambda) \\
&\quad + 2 \cdot \mathsf{Adv}^{prf}_{F,\mathcal{B}_3}(\lambda) + 3 \cdot \mathsf{Adv}^{uniq}_{F,\mathcal{B}_4}(\lambda), \\
\mathsf{Adv}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda) &\leq \mathsf{Adv}^{zk}_{\mathsf{ZKPoK},\mathsf{Sim},\mathcal{B}_1}(\lambda) + 2 \cdot \mathsf{Adv}^{prf}_{F,\mathcal{B}_2}(\lambda).
\end{aligned}$$

where n is the total number of users, and ℓ is the parameter related to $\mathcal{A}$'s output in the extended linkability game.

Proof (of Theorem 3). We prove the extended linkability, extended exculpability, and anonymity separately.

Proof (of extended linkability). The extended linkability is proved via three hybrid games.

Game G_1. This is the same as the original experiment $\mathsf{Exp}^{ex\text{-}link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$.

Game G_2. This is the same as G_1 except that upon receiving queries to $\textsc{Sign}$, we compute pid_1, pid_2 honestly but generate π via Sim, the simulator of ZKPoK. By the zero-knowledge of ZKPoK, G_2 is indistinguishable from G_1.

Game G_3. This is the same as G_2 except that we apply the extractor Ext of ZKPoK to extract the corresponding witnesses for the adversary's final output. G_3 aborts and returns 0 if $\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}_3$ holds but at least one extraction among the $\ell + 1$ signatures fails. By the strong simulation extractability of ZKPoK, G_3 and G_2 are indistinguishable.

Let $(\mathsf{e}^*, \{(R_j, \mathsf{m}_j, \sigma_j)\}_{j\in[\ell+1]}, \mu, \nu)$ be $\mathcal{A}$'s final output. Depending on the winning condition we analyze in the following two subcases.

– $(\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}_3) \wedge \mathsf{win}_4$, i.e., all $\ell + 1$ signatures are pairwise independent.
By $\mathsf{win}_1 \wedge \mathsf{win}_2$ we know all signatures are valid w.r.t. fresh ring-event-message tuples, and all zero-knowledge proofs are valid. Therefore in G_3 we can extract witnesses $\{(\delta_j, sk_{\delta_j})\}$ from the $\ell+1$ signatures such that for every $j \in [\ell + 1]$,

$$pk_{\delta_j} = F_{sk_{\delta_j}}(\alpha) \in PK_R \ \wedge \ pid_{j,1} = F_{sk_{\delta_j}}(\mathsf{e}^*||1) \ \wedge \ pid_{j,2} = F_{sk_{\delta_j}}(\mathsf{e}^*||2) + (\mathsf{m} \cdot pk_{\delta_j}).$$

By condition win_4, all signatures are pairwise independent, meaning that $\{pid_{j,1}\}_{j\in[\ell+1]}$ are all distinct. This implies that all secret keys $\{sk_{\delta_j}\}$ are distinct. Condition win_3 requires up to ℓ corruptions, meaning that there is δ_j such that $F_{sk_{\delta_j}}(\alpha) = pk = F_{sk}(\alpha)$ for some uncorrupted pk.

- If $sk_{\delta_j} \neq sk$, then $(\alpha, sk_{\delta_j}, sk)$ implies a collision of PRF, and we can construct a reduction algorithm to break the uniqueness of PRF (the second variant).

- If $sk_{\delta_j} = sk$, then we can construct a reduction algorithm that recovers an honest user's secret key sk from public key $pk = F_{sk}(\alpha)$, and hence break the security of PRF.

– $(\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}_3) \wedge (\mathsf{win}_5 \wedge \mathsf{win}_6 \wedge \mathsf{win}_7)$, i.e., two signatures σ_μ and σ_ν w.r.t. different messages are linked but failed to trace to a corrupted public key.
Let $\sigma_\mu = (pid_{\mu,1}, pid_{\mu,2}, \pi_\mu)$, $\sigma_\nu = (pid_{\nu,1}, pid_{\nu,2}, \pi_\nu)$, and let sk_{δ_μ} and sk_{δ_ν} be the extracted two secret keys. Since $\mathsf{Trace}(\mathsf{e}^*, \sigma_\mu, \sigma_\nu) = \mathsf{linked}$, we know

$$\begin{cases} pid_{\mu,1} = F_{sk_{\delta_\mu}}(\mathsf{e}^*||1) \ \wedge \ pid_{\mu,2} = F_{sk_{\delta_\mu}}(\mathsf{e}^*||2) + (\mathsf{m}_\mu \cdot pk_{\delta_\mu}), \\ pid_{\nu,1} = F_{sk_{\delta_\nu}}(\mathsf{e}^*||1) \ \wedge \ pid_{\nu,2} = F_{sk_{\delta_\nu}}(\mathsf{e}^*||2) + (\mathsf{m}_\nu \cdot pk_{\delta_\nu}), \\ pid_{\mu,1} = pid_{\nu,1}. \end{cases}$$

By win_6 it holds that $\mathsf{m}_\mu \neq \mathsf{m}_\nu$. If $sk_{\delta_\mu} = sk_{\delta_\nu}$, then $\mathsf{Trace}(\mathsf{e}^*, \sigma_\mu, \sigma_\nu)$ will return $pk_{\delta_\mu} = (\mathsf{m}_\mu - \mathsf{m}_\nu)^{-1} \cdot (pid_{\mu,2} - pid_{\nu,2})$ and hence the adversary fails. This holds even for a malicious public key $pk = E$, the additive identity element of $\mathcal{PK}$. Therefore, we must have $sk_{\delta_\mu} \neq sk_{\delta_\nu}$, and $pid_{\mu,1} = pid_{\nu,1}$ implies a collision $((\mathsf{e}^*||1), sk_\mu, sk_\nu)$ such that $F_{sk_\mu}(\mathsf{e}^*||1) = F_{sk_\nu}(\mathsf{e}^*||1)$, which breaks the uniqueness of PRF.

Taking all together concludes the proof of extended linkability. ◁

Proof (of extended exculpability). Let $((R, \mathsf{e}^*, \mathsf{m}, \sigma), (R', \mathsf{e}^*, \mathsf{m}', \sigma'))$ be the final output of $\mathcal{A}$. Following the same argument above, we first switch all zero-knowledge proofs to simulated proofs during the simulation of Sign, and then apply the extractor Ext to extract witnesses $(\delta, sk_\delta, ...)$ and $(\delta', sk_{\delta'}, ...)$. If the first signature was created by the signing oracle, then only the second witness $(\delta', sk_{\delta'}, ...)$ is extracted. There are two ways for $\mathcal{A}$ to win.

– $\mathsf{win}_3 \wedge \mathsf{win}_4$, i.e., $\mathcal{A}$ forges one signature that together with another honestly or adversarially generated signature trace to one honest user with pk.
– $\mathsf{win}_5 \wedge \mathsf{win}_6$, i.e., $\mathcal{A}$ forges one signature that is linked to an existing signature generated by an honest user with public key pk_δ.

Let $\sigma = (pid_1, pid_2, \pi)$ and $\sigma' = (pid'_1, pid'_2, \pi')$. We have $pid_1 = pid'_1$, and additionally

$$\begin{cases} pid_1 = F_{sk_\delta}(\mathsf{e}^*||1) \ \wedge \ pid_2 = F_{sk_\delta}(\mathsf{e}^*||2) + (\mathsf{m} \cdot pk_\delta), \\ pid'_1 = F_{sk_{\delta'}}(\mathsf{e}^*||1) \ \wedge \ pid'_2 = F_{sk_{\delta'}}(\mathsf{e}^*||2) + (\mathsf{m}' \cdot pk_{\delta'}). \end{cases}$$

We first consider the case that $(R, \mathsf{e}^*, \mathsf{m}, \sigma)$ was from the signing oracle. Let sk_δ be the secret key of the corresponding honest signer, and $sk_{\delta'}$ be the extracted key from $\mathcal{A}$'s output.

– Case 1.1. $sk_{\delta'} \neq sk_\delta$. In this case $pid_1 = F_{sk_\delta}(\mathsf{e}^*||1) = F_{sk_{\delta'}}(\mathsf{e}^*||1) = pid'_1$ implies a collision $((\mathsf{e}^*||1), sk_\delta, sk_{\delta'})$ of PRF, and we can construct a reduction algorithm to break uniqueness of PRF (the first variant).
– Case 1.2. $sk_{\delta'} = sk_\delta$. In this case we can construct a reduction algorithm to recover the secret key sk_δ of an honest user, which breaks the security of PRF.

We now consider the case that both $(R, \mathsf{e}^*, \mathsf{m}, \sigma)$ and $(R', \mathsf{e}^*, \mathsf{m}', \sigma')$ are adversarially forgeries. Let sk_δ and sk'_δ be the extracted secret keys, respectively.

– Case 2.1. $sk_\delta \neq sk'_\delta$. In this case $pid_1 = F_{sk_\delta}(\mathsf{e}^*||1) = F_{sk_{\delta'}}(\mathsf{e}^*||1) = pid'_1$ implies a collision $((\mathsf{e}^*||1), sk_\delta, sk_{\delta'})$ of PRF, and we can construct a reduction algorithm to break uniqueness of PRF.
– Case 2.2. $sk_\delta = sk'_\delta$. Then $pid_1 = pid'_1$. To win the extended exculpability game, we require $\mathsf{m} \neq \mathsf{m}'$, and $(\mathsf{m} - \mathsf{m}')^{-1} \cdot (pid_2 - pid'_2) = pk$ for some uncorrupted pk. Let sk be the corresponding secret key of pk.
  - Case 2.2.1. $sk \neq sk_\delta$. Since $pid_2 = F_{sk_\delta}(\mathsf{e}^*||2) + (\mathsf{m} \cdot pk_\delta)$ and $pid'_2 = F_{sk_\delta}(\mathsf{e}^*||2) + (\mathsf{m}' \cdot pk_\delta)$, we have $pk = pk_\delta$, i.e., $F_{sk}(\alpha) = F_{sk_\delta}(\alpha)$. Then, we can construct a reduction algorithm that uses the collision (α, sk, sk_δ) to break the uniqueness of PRF (the second variant).
  - Case 2.2.2. $sk = sk_\delta$. This implies the recovering of sk for an honest user, which breaks the security of PRF. ◁

Proof (of Anonymity). The anonymity is proved via three hybrid games.

Game G_1**.** This is the same as the original experiment $\mathsf{Exp}^{anony}_{\mathsf{TRS},\mathcal{A}}(\lambda)$.

Game G_2. This is the same as G_1 except that upon receiving queries to SIGN, we compute pid_1, pid_2 honestly but generate π via Sim, the simulator of ZKPoK. By the zero-knowledge of ZKPoK, G_2 is indistinguishable from G_1.

Game G_3. In this game, we switch the computation of $F_{sk_1}()$ and $F_{sk_2}()$ of the first and the second users to truly random functions $R_1()$ and $R_2()$. Namely, the experiment now maintains two lists R_1 and R_2. Let $b \in \{1, 2\}$. Whenever there is a need to compute $F_{sk_b}(x)$ (including the computation of the public keys), the experiment returns $\mathsf{R}_b[x]$ directly if it exists, otherwise it randomly samples $y \leftarrow \mathcal{PK}$ (recall that the image space of F equals the public key space), updates $\mathsf{R}_b[x] \leftarrow y$ and returns y. In G_3, the experiment does not need the secret keys to simulate the signing oracle for the 1-st and the 2-nd users. Thanks to the security of PRF, G_3 and G_2 are indistinguishable.

Let $b \in \{1, 2\}$ be the challenge bit. For a specific query to $\textsc{Chall}(R, \mathsf{e}^*, \mathsf{m})$, the returned signature is of the form

$$\sigma = (pid_1, pid_2, \pi) = (y_{b,1}, y_{b,2} + \mathsf{m} \cdot pk_b, \pi),$$

where $y_{b,1}$ and $y_{b,2}$ are random values sampled from $\mathcal{PK}$. Due to the restriction of G_3, the adversary is not allowed to query $\textsc{Sign}(b \in \{1, 2\}, R, \mathsf{e}^*, \mathsf{m})$ or $\textsc{Chall}(R, \mathsf{e}^*, \mathsf{m}' \neq \mathsf{m})$. In other words, $y_{b,2}$ is only used once to mask $\mathsf{m} \cdot pk_b$. Due to the perfect security of the one-time pad, the adversary has no advantage to deduce b in G_3, which concludes the proof of anonymity. ◁

□

Strong Unforgeability. It is easy to see that our scheme also achieves strong unforgeability, if the underlying ZKPoK scheme has strong simulation extractability. Our instantiation in Sec. 5 achieves this property.

5 Concrete Construction from Bulletproofs in the Random Oracle Model

5.1 Instantiation of PRF in the ROM

Let $H : \{0,1\}^* \to \mathbb{G}$ be a hash function modeled as a random oracle. We instantiate PRF with key space $\mathcal{K} = \mathbb{Z}_q$, input space $\mathcal{X} = \{0,1\}^*$, and output space $\mathcal{Y} = \mathbb{G}$ as follows.

$$F_{sk}(x) = (H(x))^{sk}.$$

Now we prove the security and uniqueness.

Theorem 4. *Assume H is a random oracle and the DDH assumption holds over $\mathbb{G}$, then the PRF constructed above is secure (pseudorandom) and unique.*

Proof (of Theorem 4). We first prove the pseudorandomness based on the DDH assumption and the random oracle hypothesis. Let Q denote the total number of queries by the adversary $\mathcal{A}$. Therefore, $\mathcal{A}$'s goal is to distinguish the following two distributions,

$$\begin{pmatrix} g_1 & g_2 & \cdots & g_Q \\ g_1^k & g_2^k & \cdots & g_Q^k \end{pmatrix} \text{ and } \begin{pmatrix} g_1 & g_2 & \cdots & g_Q \\ g'_1 & g'_2 & \cdots & g'_Q \end{pmatrix}$$

where $k \leftarrow \mathbb{Z}_q$ denotes the secret key, $H(x_i) = g_i$ for $i \in [Q]$, and $g'_1, g'_2, ..., g'_Q \xleftarrow{\$} \mathbb{G}$. The left distribution corresponds to the view of the real PRF, while the right corresponds to the view of the truly random function. By the DDH assumption (in fact the Q-DDH assumption, see Appendix A), the two distributions are indistinguishable.

Let $1_{\mathbb{G}}$ be the identity element of $\mathbb{G}$. For any adversary making at most Q_H queries to H, we have

$$\begin{aligned}\mathsf{Adv}^{uniq}_{F,\mathcal{A}}(\lambda) &:= \Pr[(x, sk_1, sk_2) \leftarrow \mathcal{A} \ : \ sk_1 \neq sk_2 \wedge H(x)^{sk_1} = H(x)^{sk_2}] \\ &= \Pr[(x, sk_1, sk_2) \leftarrow \mathcal{A} \ : \ H(x) = 1_{\mathbb{G}}] \leq \frac{Q_H}{q}.\end{aligned}$$

This finishes the proof of uniqueness. □

In the construction of TRS, the public key is set to be $pk = F_{sk}(\alpha) = g_\alpha^{sk}$, where $g_\alpha = H(\alpha)$. In fact, the choice of α does not affect the security of TRS at all. From now on we simply assume $g = H(\alpha)$ to keep notations simple, and hence $pk = g^{sk}$.

5.2 Instantiation of ZKPoK with Bulletproofs in the ROM

Based on the PRF above, a pair of keys is of the form $(pk, sk) = (X = g^x, x)$. Let $PK_R = (X_1, ..., X_n) \in \mathbb{G}^n$ be a set of public keys. Let $\mathsf{m} \in \mathbb{Z}_q$ be the message to be signed, and let $(g_{\mathsf{e},1}, g_{\mathsf{e},2}) = (H(\mathsf{e}||1), H(\mathsf{e}||2)) \in \mathbb{G}^2$ be the hash outputs of an event label $\mathsf{e} \in \{0,1\}^*$. Now we are going to construct a ZKPoK scheme for the following language:

$$\begin{aligned}\mathcal{L}_{\mathsf{DL\text{-}TRS}} := \{ins = (X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2) \mid \exists wit = (\delta, x) \ s.t. \ \delta \in [n] \ \wedge \ X_\delta = g^x \\ \wedge \ pid_1 = g_{\mathsf{e},1}^x \ \wedge \ pid_2 = g_{\mathsf{e},2}^x \cdot g^{x \cdot \mathsf{m}}\}.\end{aligned}$$

Following the notations in [BBB+18], we use bold font to denote vectors, i.e., $\mathbf{a} = (a_1, ..., a_n) \in \mathbb{F}^n$. For $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$, denote by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i \cdot b_i \in \mathbb{F}$ the inner product of $\mathbf{a}, \mathbf{b}$, and by $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n) \in \mathbb{F}^n$ the Hadamard product of $\mathbf{a}, \mathbf{b}$. For $a \in \mathbb{Z}_q$ and vectors $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}^n$, $\mathbf{a} = (a_1, ..., a_n) \in \mathbb{Z}_q^n$, denote $\mathbf{g}^a = (g_1^a, g_2^a, ..., g_n^a) \in \mathbb{G}^n$, and $\mathbf{g}^{\mathbf{a}} = \prod_{i=1}^n g_i^{a_i} \in \mathbb{G}$. To get the high-level idea of the design, in the following we describe the ZKPoK scheme in the interactive protocol setting. The non-interactive proof can be achieved via the Fiat-Shamir paradigm assuming random oracles.

The common reference string consists of a set of random generators $\mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{G}^n$ and $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{G}^n$. Let $H' : \{0,1\}^* \to \mathbb{G}$ be a hash function modeled as a random oracle and $\mathbf{pk} = (X_1, \ldots, X_n) \in \mathbb{G}^n$ under a canonical encoding, the blinding generator is derived as

$$\tilde{g} := H'(\mathbf{pk}) \in \mathbb{G}.$$

Define $\mathbf{u} = \mathbf{pk}^w \circ \mathbf{p}$, where $w \in \mathbb{Z}_q$ is a random challenge. As shown in [GK15], $\mathbf{u}$ is uniformly distributed if $\mathbf{p}$ is.

Remark 2. It is essential to derive $\tilde{g}$ from the ring, rather than from the event label or CRS. If $\tilde{g}$ were fixed in the CRS, or set to $H'(\mathsf{e})$, an adversary could first obtain $\tilde{g}$ and only then register the public key $X_\delta := g^x \tilde{g}^d$ for known x, d, and then mounts a rogue-key attack against the extended linkability, as discussed in Sec. 1.3.

Log-Size Proof for the Binary Vector. We first show how to obtain a log-size proof for the secret index δ, following the same method in [BBB+18]. Let $\mathbf{a}_L = (a_1, ..., a_n) \in \{0,1\}^n$ such that $a_\delta = 1$ and $a_{i \neq \delta} = 0$. The proof of $\mathbf{a}_L$'s well-formedness can be transferred into proving the knowledge of $\mathbf{a}_L$ such that

$$\langle \mathbf{a}_L, \mathbf{1} \rangle = 1 \ \wedge \ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0} \ \wedge \ \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}.$$

The prover can prove the above equations by proving that

$$\langle \mathbf{a}_L - z \cdot \mathbf{1} \ , \ \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}) + z^2 \cdot \mathbf{1} \rangle = z^2 + \Phi(y, z),$$

where $y, z \in \mathbb{Z}_q$ are randomly sampled, $\mathbf{y}^n = (1, y, y^2, ..., y^{n-1}) \in \mathbb{Z}_q^n$, and $\Phi(y, z) = (z - z^2) \cdot \langle \mathbf{1}, \mathbf{y}^n \rangle - z^3 \cdot \langle \mathbf{1}, \mathbf{1} \rangle \in \mathbb{Z}_q$.

Let $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$ be a commitment of $\mathbf{a}_L, \mathbf{a}_R$ with randomness $r_A \in \mathbb{Z}_q$. The prover now randomly samples $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_q^n$ and gets commitment $S = \mathbf{u}^{\mathbf{s}_L}\mathbf{v}^{\mathbf{s}_R}\tilde{g}^{r_S}$. After seeing A, S, the verifier returns random challenges $y, z \xleftarrow{\$} \mathbb{Z}_q$, and the prover is able to compute the following vector polynomials:

$$\begin{array}{ll} l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}) + \mathbf{s}_L \cdot X & \in \mathbb{Z}_q^n[X] \\ r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1} + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{1} & \in \mathbb{Z}_q^n[X] \\ t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2 & \in \mathbb{Z}_q[X] \end{array}$$

where

$$t_0 = z^2 + \Phi(y, z).$$

The prover now commits to the remaining coefficients $t_1, t_2 \in \mathbb{Z}_q$, and then convinces the verifier the above equations hold. This is done by checking the value of $t(X)$ at a random point $p \in \mathbb{Z}_q^*$. Specifically, the prover sends commitments $T_1 = g^{t_1}\tilde{g}^{r_{T,1}}$ and $T_2 = g^{t_2}\tilde{g}^{r_{T,2}}$ to the verifier. After receiving a random challenge $x \xleftarrow{\$} \mathbb{Z}_q^*$, the prover computes:

$$\begin{array}{ll} \mathbf{l} = l(p) = (\mathbf{a}_L - z \cdot \mathbf{1}) + \mathbf{s}_L \cdot p & // \text{ value of } l(X) \text{ on the point } p \\ \mathbf{r} = r(p) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1} + \mathbf{s}_R \cdot p) + z^2 \cdot \mathbf{1} & // \text{ value of } r(X) \text{ on the point } p \\ \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle & // \ t = \langle \mathbf{l}, \mathbf{r} \rangle \\ r_{\hat{t}} = r_{T,1} \cdot p + r_{T,2} \cdot p^2 & // \text{ blinding value for } \hat{t} \\ r' = r_A + r_S \cdot p & // \text{ new blinding value for commitment to } \mathbf{l}, \mathbf{r} \end{array}$$

The prover sends $(\mathbf{l}, \mathbf{r}, r_{\hat{t}}, r', \hat{t})$ to the verifier. Then the verifier computes and checks:

$$\begin{array}{ll} \mathbf{v}' := \mathbf{v}^{(\mathbf{y}^{-n})} \in \mathbb{G}^n & // \ \mathbf{v}' = (v_1, v_2^{y^{-1}}, v_3^{y^{-2}}, ..., v_n^{y^{-n+1}}) \\ g^{\hat{t}}\tilde{g}^{r_{\hat{t}}} = g^{z^2 + \Phi(y,z)} \cdot T_1^p \cdot T_2^{p^2} & // \text{ check that } \hat{t} = t(p) = t_0 + t_1 p + t_2 p^2 \\ A' := A \cdot S^p \cdot \mathbf{u}^{-z} \cdot (\mathbf{v}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{1}} & // \text{ compute the commitment to } l(p), r(p) \text{ from } A, S \\ A' \stackrel{?}{=} \mathbf{u}^{\mathbf{l}} \cdot (\mathbf{v}')^{\mathbf{r}} \cdot \tilde{g}^{r'} & // \text{ check that } \mathbf{l}, \mathbf{r} \text{ are correct} \\ \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle & // \text{ check the correctness of } \hat{t} \end{array}$$

The proof size is dominated by $\mathbf{l}, \mathbf{r} \in \mathbb{Z}_q^n$. To use the log-size inner-product argument of Bulletproofs, instead of sending $(\mathbf{l}, \mathbf{r}, r_{\hat{t}}, r', \hat{t})$, now the prover sends only $(r_{\hat{t}}, r', \hat{t})$, and run the inner-product protocol with the verifier on public input $(\mathbf{u}, \mathbf{v}', A'\tilde{g}^{-r'}, \hat{t})$, to prove that

$$A'\tilde{g}^{-r'} = \mathbf{u}^{\mathbf{l}} \cdot (\mathbf{v}')^{\mathbf{r}} \ \wedge \ \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle.$$

Via a recursive use of the inner-product argument, the communication complexity can be reduced to $O(\log n)$. Note that the vectors $\mathbf{l}, \mathbf{r}$ do not leak any information on $\mathbf{a}_L, \mathbf{a}_R$ due to the random blinding vectors $\mathbf{s}_L, \mathbf{s}_R$.

Knowledge of the Secret Key and Well-Formedness of pid. The above argument proves that the vector $\mathbf{a}_L$ in the committed in $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$ satisfies that $\mathbf{a}_L \in \{0,1\}^n$ and the Hamming weight of $\mathbf{a}_L$ is 1. Now we bind this with the knowledge of the secret key and the well-formedness of pid_1, pid_2, completing the proof for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$.

Let $pk = g^x$ be the public key with the corresponding secret key x. Therefore, $\mathbf{pk}^{\mathbf{a}_L} = pk = g^x$. Let $C = pk \cdot \tilde{g}^{r_C}$ be a commitment to the public key, and $\hat{A} = \mathbf{p}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{\hat{r}_A}$ be a commitment of $\mathbf{a}_L$ and $\mathbf{a}_R$ with randomness $\hat{r}_A \in \mathbb{Z}_q$. For any $w \in \mathbb{Z}_q$, if $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$ and $r_A = r_C \cdot w + \hat{r}_A$, then

$$\begin{aligned} A &= (\mathbf{pk}^w \circ \mathbf{p})^{\mathbf{a}_L} \cdot \mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A} \\ &= (\mathbf{pk}^{\mathbf{a}_L})^w \cdot \mathbf{p}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A} \\ &= pk^w \cdot \mathbf{p}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_C \cdot w + \hat{r}_A} \\ &= (pk \cdot \tilde{g}^{r_C})^w \cdot \mathbf{p}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{\hat{r}_A} \\ &= C^w \cdot \hat{A}. \end{aligned}$$

Therefore, to convince that $\mathbf{a}_L$ in the commitment $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$ and pk in the commitment $C = g^x\tilde{g}^{r_C}$ satisfy $\mathbf{pk}^{\mathbf{a}_L} = pk = g^x$, the prover first sends $\hat{\mathbf{A}}, C$ to the verifier. Upon receiving a random challenge $w \xleftarrow{\$} \mathbb{Z}_q$, the prover computes $\mathbf{u} = \mathbf{pk}^w \circ \mathbf{p}$ and $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$, and shows that

$$A = C^w \cdot \hat{A}.$$

The only thing left now is to prove the well-formedness of pid_1 and pid_2 w.r.t. the secret key x committed in $C = g^x\tilde{g}^{r_C}$, which can be done via the Schnorr proof of knowledge. Namely, the prover sends $D = g^r\tilde{g}^{r_D}$, the commitment of r, and $C_1 = g_{\mathsf{e},1}^r$, $C_2 = g_{\mathsf{e},2}^r \cdot g^{r\cdot\mathsf{m}}$ to the verifier. Given random challenge $w \in \mathbb{Z}_q$ (we can reuse w here), the prover sends

$$s = r - x \cdot w \text{ and } s_D = r_D - r_C \cdot w$$

to the verifier. Then the verifier checks if

$$\begin{array}{ll} D = C^w \cdot g^s \cdot \tilde{g}^{s_D} & \text{// check the correctness of } s = r - x\cdot w \text{ from commitments } C, D \\ C_1 = pid_1^w \cdot g_{\mathsf{e},1}^s & \text{// check } pid_1 = g_{\mathsf{e},1}^x \\ C_2 = pid_2^w \cdot g_{\mathsf{e},2}^s \cdot g^{s\cdot\mathsf{m}} & \text{// check } pid_2 = g_{\mathsf{e},2}^x \cdot g^{x\cdot\mathsf{m}} \end{array}$$

Taking All Together. Let $\Pi_{\mathsf{IP}} = (\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Ver})$ be a non-interactive proof system for the relation:

$$\mathcal{L}_{\mathsf{IP}} := \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_q; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_q) : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \ \wedge \ c = \langle \mathbf{a}, \mathbf{b}\rangle\}.$$

Let $\mathbb{G}$ be a cyclic group of prime order q with g a generator. Let $\{H^b : \{0,1\}^* \to \mathbb{Z}_q^b\}_{b\in\{1,2\}}$ be a family of hash functions. Assume the size of the anonymous ring is upper bounded by $n \in \mathbb{Z}^+$. Our ZKPoK for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$ is shown in Fig. 6. As all Sigma-protocol-based signature schemes, we include the hash values into the final signature rather than the commitments to save signature size.

We provide the concrete ZKPoK construction for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$ in Fig. 6.

Theorem 5. *If the DLog assumption holds on $\mathbb{G}$ and Π_{IP} is an NIZK scheme with completeness, zero-knowledge and witness-extended emulation [BBB+18], and H^1, H^2 are modeled as random oracles, then the ZKPoK scheme in Fig. 6 has completeness, zero-knowledge and strong simulation extractability.*

The proof is deferred to Appendix C.

5.3 Concrete Signature Size

The proof for $\mathcal{L}_{\mathsf{IP}}$ consists of $2\log n$ elements in $\mathbb{G}$ and 2 elements in $\mathbb{Z}_q$. Consequently, the ZKPoK scheme in Fig. 6 has a proof size $(2\log n + 4)\cdot|\mathbb{G}| + 11\cdot|\mathbb{Z}_q|$, making the TRS scheme have a signature size $(2\log n + 6)\cdot|\mathbb{G}| + 11\cdot|\mathbb{Z}_q|$.

When instantiated on Curve25519, where both elements of $\mathbb{Z}_q$ and $\mathbb{G}$ take 32 bytes, the resulting concrete signature size becomes $(64\cdot\log n + 544)$ bytes. A comparison with other schemes in terms of concrete signature size is provided in Table 2.

Acknowledgments. We would like to thank Daniel Collins for inspiring this project and for many valuable discussions.

References

[ABF25] Gennaro Avitabile, Vincenzo Botta, and Dario Fiore. Tetris! traceable extendable threshold ring signatures and more. *Cryptology ePrint Archive*, 2025.

[ALSY13] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theoretical Computer Science*, 469:1–14, 2013.

$\underline{\mathsf{GenCRS}(1^\lambda)}$

1. $\mathbf{p}, \mathbf{v} \xleftarrow{\$} \mathbb{G}^n$
2. **return** $\mathsf{crs} := (\mathbf{p}, \mathbf{v})$

$\underline{\mathsf{Prove}(\mathsf{crs}, ins, wit)}$

3. $(X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2) \leftarrow ins$
4. $\tilde{g} := H'(X_1, ..., X_n)$
5. $(\delta \in [n], x \in \mathbb{Z}_q) \leftarrow wit$
6. $\mathsf{info} := ins$ // the up-to-now public information
7. **for** $i \in [n] \setminus \{\delta\} : a_i := 0$
8. $a_\delta := 1$
9. $\mathbf{a}_L := (a_1, ..., a_n) \in \{0,1\}^n \subseteq \mathbb{Z}_q^n$; $\mathbf{a}_R := \mathbf{a}_L - \mathbf{1} \in \mathbb{Z}_q^n$
10. $\hat{r}_A \xleftarrow{\$} \mathbb{Z}_q$; $\hat{A} := \mathbf{p}^{\mathbf{a}_L} \mathbf{v}^{\mathbf{a}_R} \tilde{g}^{\hat{r}_A}$
11. $r_C \xleftarrow{\$} \mathbb{Z}_q$; $C := g^x \cdot \tilde{g}^{r_C}$
12. $r, r_D \xleftarrow{\$} \mathbb{Z}_q$; $D := g^r \tilde{g}^{r_D}$
13. $C_1 := g_{\mathsf{e},1}^r$; $C_2 := g_{\mathsf{e},2}^r \cdot g^{r \cdot \mathsf{m}}$
14. $\mathsf{info} := \mathsf{info} || (\hat{A}, C, D, C_1, C_2)$; $w := H^1(\mathsf{info}) \in \mathbb{Z}_q$
15. $s = r - x \cdot w$; $s_D = r_D - r_C \cdot w$
16. $\mathbf{u} := (X_1, ..., X_n)^w \circ \mathbf{p} \in \mathbb{G}^n$
17. $r_A := r_C \cdot w + \hat{r}_A$; $A := \mathbf{u}^{\mathbf{a}_L} \mathbf{v}^{\mathbf{a}_R} \tilde{g}^{r_A}$
18. $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_q^n$; $r_S \xleftarrow{\$} \mathbb{Z}_q$
19. $S := \mathbf{u}^{\mathbf{s}_L} \mathbf{v}^{\mathbf{s}_R} \tilde{g}^{r_S}$
20. $\mathsf{info} := \mathsf{info} || (A, S, s, s_D)$; $(y, z) := H^2(\mathsf{info}) \in \mathbb{Z}_q^2$
21. $l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}) + \mathbf{s}_L \cdot X \in \mathbb{Z}_q^n[X]$
22. $r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1} + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{1} \in \mathbb{Z}_q^n[X]$
23. $t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2 \in \mathbb{Z}_q[X]$
24. $r_{T,1}, r_{T,2} \xleftarrow{\$} \mathbb{Z}_q$
25. $T_1 := g^{t_1} \cdot \tilde{g}^{r_{T,1}}$; $T_2 := g^{t_2} \cdot \tilde{g}^{r_{T,2}}$
26. $\mathsf{info} := \mathsf{info} || (T_1, T_2)$; $p := H^1(\mathsf{info})$
27. **if** $p = 0$: **rerun** $\mathsf{Prove}(\mathsf{crs}, ins, wit)$
28. // require $p \in \mathbb{Z}_q^*$
29. $r_{\hat{t}} := r_{T,1} \cdot p + r_{T,2} \cdot p^2$
30. $r' := r_A + r_S \cdot p$
31. $\mathbf{v}' := \mathbf{v}^{(\mathbf{y}^{-n})}$
32. $\mathbf{l} := l(p) \in \mathbb{Z}_q^n$; $\mathbf{r} := r(p) \in \mathbb{Z}_q^n$; $\hat{t} := \langle \mathbf{l}, \mathbf{r} \rangle$
33. $\pi_{\mathsf{IP}} \xleftarrow{\$} \Pi_{\mathsf{IP}}.\mathsf{Prove}((\mathbf{u}, \mathbf{v}', \mathbf{u}^{\mathbf{l}}(\mathbf{v}')^{\mathbf{r}}, \hat{t}), (\mathbf{l}, \mathbf{r}))$
34. $\pi := (\hat{A}, C, S, s, s_D, T_2, r_{\hat{t}}, r', \hat{t}, w, y, z, p, \pi_{\mathsf{IP}})$
35. **return** π

$\underline{\mathsf{Ver}(\mathsf{crs}, ins, \pi)}$

36. $(X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2) \leftarrow ins$
37. $\tilde{g} := H'(X_1, ..., X_n)$
38. $(\hat{A}, C, S, s, s_D, T_2, r_{\hat{t}}, r', \hat{t}, w, y, z, p, \pi_{\mathsf{IP}}) \leftarrow \pi$
39. $A := C^w \cdot \hat{A}$; $D := C^w \cdot g^s \cdot \tilde{g}^{s_D}$
40. $C_1 := pid_1^w \cdot g_{\mathsf{e},1}^s$; $C_2 := pid_2^w \cdot g_{\mathsf{e},2}^s \cdot g^{s \cdot \mathsf{m}}$
41. $T_1 := (g^{\hat{t} - z^2 - \Phi(y,z)} \tilde{g}^{r_{\hat{t}}} T_2^{-p^2})^{p^{-1}}$
42. $\mathsf{info}_1 := (ins || (\hat{A}, C, D, C_1, C_2))$
43. $\mathsf{info}_2 := (\mathsf{info}_1 || (A, S, s, s_D))$
44. $\mathsf{info}_3 := (\mathsf{info}_2 || (T_1, T_2))$
45. $\mathbf{y} := (1, y, y^2, ..., y^{n-1}) \in \mathbb{Z}_q^n$
46. $\mathbf{u} := (X_1, ..., X_n)^w \circ \mathbf{p} \in \mathbb{G}_n$
47. $\mathbf{v}' := \mathbf{v}^{\mathbf{y}^{-n}} \in \mathbb{G}^n$
48. $A' := A \cdot S^p \cdot \mathbf{u}^{-z} (\mathbf{v}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{1}} \cdot \tilde{g}^{-r'}$
49. // recompute the commitment $\mathbf{u}^{\mathbf{l}} \cdot (\mathbf{v}')^{\mathbf{r}}$
50. $\Phi(y, z) := (z - z^2) \cdot \langle \mathbf{1}, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}, \mathbf{1} \rangle \in \mathbb{Z}_q$
51. $b_1 := [\![w = H^1(\mathsf{info}_1)]\!]$
52. $b_2 := [\![(y, z) = H^2(\mathsf{info}_2)]\!]$
53. $b_3 := [\![p = H^1(\mathsf{info}_3)]\!]$
54. $b_4 \leftarrow \Pi_{\mathsf{IP}}.\mathsf{Ver}((\mathbf{u}, \mathbf{v}', A', \hat{t}), \pi_{\mathsf{IP}})$
55. **if** $(b_1 \wedge b_2 \wedge b_3 \wedge b_4)$: **return** 1
56. **else** : **return** 0

$\underline{\mathsf{Sim}(\mathsf{crs}, ins)}$

57. Sim can reprogram random oracles H^1, H^2
58. $(X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2) \leftarrow ins$
59. $\tilde{g} := H'(X_1, ..., X_n)$
60. $\mathbf{l}, \mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$; $\hat{t} := \langle \mathbf{l}, \mathbf{r} \rangle$; $p, w, y, z \xleftarrow{\$} \mathbb{Z}_q$
61. $\Phi(y, z) := (z - z^2) \cdot \langle \mathbf{1}, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}, \mathbf{1} \rangle \in \mathbb{Z}_q$
62. $\mathbf{y} := (1, y, y^2, ..., y^{n-1}) \in \mathbb{Z}_q^n$
63. $\mathbf{u} := (X_1, ..., X_n)^w \circ \mathbf{p} \in \mathbb{G}_n$; $\mathbf{v}' := \mathbf{v}^{\mathbf{y}^{-n}} \in \mathbb{G}^n$
64. $\pi_{\mathsf{IP}} \xleftarrow{\$} \Pi_{\mathsf{IP}}.\mathsf{Prove}((\mathbf{u}, \mathbf{v}', \mathbf{u}^{\mathbf{l}}(\mathbf{v}')^{\mathbf{r}}, \hat{t}), (\mathbf{l}, \mathbf{r}))$
65. $r_{\hat{t}}, r', s, s_D \xleftarrow{\$} \mathbb{Z}_q$
66. $T_2 \xleftarrow{\$} \mathbb{G}$; $T_1 := (g^{\hat{t} - z^2 - \Phi(y,z)} \cdot \tilde{g}^{r_{\hat{t}}} \cdot T_2^{-p^2})^{p^{-1}}$
67. $A' := \mathbf{u}^{\mathbf{l}} (\mathbf{v}')^{\mathbf{r}}$
68. $S \xleftarrow{\$} \mathbb{G}$; $A := A' \cdot S^{-p} \cdot \mathbf{u}^z (\mathbf{v}')^{-\mathbf{z} \cdot \mathbf{y}^n - z^2 \cdot \mathbf{1}} \cdot \tilde{g}^{r'}$
69. $C \xleftarrow{\$} \mathbb{G}$; $\hat{A} := A \cdot C^{-w}$; $D := C^w \cdot g^s \cdot \tilde{g}^{s_D}$
70. $C_1 := pid_1^w \cdot g_{\mathsf{e},1}^s$; $C_2 := pid_2^w \cdot g_{\mathsf{e},2}^s \cdot g^{s \cdot \mathsf{m}}$
71. $\mathsf{info}_1 := (ins || (\hat{A}, C, D, C_1, C_2))$
72. $\mathsf{info}_2 := (\mathsf{info}_1 || (A, S, s, s_D))$
73. $\mathsf{info}_3 := (\mathsf{info}_2 || (T_1, T_2))$
74. **reprogram** $H^1(\mathsf{info}_1) = w$
75. **reprogram** $H^2(\mathsf{info}_2) = (y, z)$
76. **reprogram** $H^1(\mathsf{info}_3) = p$
77. **if** any **reprogram** fails: **return** $\perp$
78. **return** $\pi := (\hat{A}, C, S, s, s_D, T_2, r_{\hat{t}}, r', \hat{t}, w, y, z, p, \pi_{\mathsf{IP}})$

Fig. 6. ZKPoK construction for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$ and the corresponding simulator Sim. For $y, z \in \mathbb{Z}_q$, define $\Phi(y, z) = (z - z^2) \cdot \langle \mathbf{1}, \mathbf{y}^n \rangle - z^3 \cdot \langle \mathbf{1}, \mathbf{1} \rangle \in \mathbb{Z}_q$.

Table 2. Comparison of concrete signature sizes (in bytes) across TRS schemes. Here n denotes the ring size, and m, n', q, p, q_1 denote parameters in the lattice-based constructions in [FLL+21, YLGT23]. Some entries are left blank due to absent data or unspecified parameters in the cited references.

Scheme & Concrete/Asymptotic Complexity	**Ring Size**									
	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
FS07 [FS07] $64 \cdot n + 32$ $(2n \cdot \lvert\mathbb{Z}_q\rvert + \lvert\mathbb{G}\rvert)$	160	288	544	1056	2080	4128	8224	16416	32800	65568
FLWL20 [FLWL20, FLL+21] (Lattice) $O(\log n \cdot (n' \log q_1) + m \cdot \log^2 pq)$										
FLL+21 [FLL+21] (Symmetric) $232632 \cdot \log n + 908959$	1141591	1374223	1606855	1839487	2072119	2304751	2537383	2770015	3002647	3235279
YLGT23 [YLGT23] $25.36536 \cdot n + 7444.48$ $(n \cdot d \log 3 + n' d \log q + md \log(2md^2))$	7497	7548	7650	7847	8256	9070	10696	13938	20433	33428
WLB+24-ISO [WLB+24]	4556	6584	8448	10332	12355	14199				22732
WLB+24-LAT [WLB+24]	57708	58732	59756	60780	61804	62828				66867
This work $64 \cdot \log n + 544$ $((2 \log n + 6)\lvert\mathbb{G}\rvert + 11 \cdot \lvert\mathbb{Z}_q\rvert)$	608	672	736	800	864	928	992	1056	1120	1184

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Berlin, Heidelberg, April 2015.

[ASY06] Man Ho Au, Willy Susilo, and Siu-Ming Yiu. Event-oriented k-times revocable-iff-linked group signatures. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 06*, volume 4058 of *LNCS*, pages 223–234. Springer, Berlin, Heidelberg, July 2006.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[BEHM22] Jonathan Bootle, Kaoutar Elkhiyaoui, Julia Hesse, and Yacov Manevich. DualDory: Logarithmic-verifier linkable ring signatures through preprocessing. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part II*, volume 13555 of *LNCS*, pages 427–446. Springer, Cham, September 2022.

[BGK+21] Pourandokht Behrouz, Panagiotis Grontas, Vangelis Konstantakatos, Aris Pagourtzis, and Marianna Spyrakou. Designated-verifier linkable ring signatures. In *International Conference on Information Security and Cryptology*, pages 51–70. Springer, 2021.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Cham, December 2020.

[BL16] Xavier Bultel and Pascal Lafourcade. k-times full traceable ring signature. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 39–48. IEEE, 2016.

[BM18] Pedro Branco and Paulo Mateus. A code-based linkable ring signature scheme. In Joonsang Baek, Willy Susilo, and Jongkil Kim, editors, *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219. Springer, Cham, October 2018.

[BM19] Pedro Branco and Paulo Mateus. A traceable ring signature scheme based on coding theory. In Jintai Ding and Rainer Steinwandt, editors, *PQCrypto 2019*, pages 387–403. Springer, Cham, 2019.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Berlin, Heidelberg, May 2005.

[CLW08] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *NDSS 2008*. The Internet Society, February 2008.

[CLXZ25] Wonseok Choi, Xiangyu Liu, Lirong Xia, and Vassilis Zikas. K-linkable ring signatures and applications in generalized voting. Cryptology ePrint Archive, Report 2025/243, 2025.

[DRS18] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *PQCrypto 2018*, pages 419–440. Springer, Cham, 2018.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013.

[ESS+19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 2019*, volume 11464 of *LNCS*, pages 67–88. Springer, Cham, June 2019.

[EZS+19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

[FLL+21] Hanwen Feng, Jianwei Liu, Dawei Li, Ya-Nan Li, and Qianhong Wu. Traceable ring signatures: general framework and post-quantum security. *DCC*, 89(6):1111–1145, 2021.

[FLWL20] Hanwen Feng, Jianwei Liu, Qianhong Wu, and Ya-Nan Li. Traceable ring signatures with post-quantum security. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 442–468. Springer, Cham, February 2020.

[FQ21] Ashley Fraser and Elizabeth A Quaglia. Report and trace ring signatures. In *International Conference on Cryptology and Network Security*, pages 179–199. Springer, 2021.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 181–200. Springer, Berlin, Heidelberg, April 2007.

[Fuj11] Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 393–415. Springer, Berlin, Heidelberg, February 2011.

[GDW20] Ke Gu, Xinying Dong, and Linyu Wang. Efficient traceable ring signature scheme without pairings. *Advances in Mathematics of Communications*, 14(2):207–232, 2020.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Berlin, Heidelberg, May 2016.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 253–280. Springer, Berlin, Heidelberg, April 2015.

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-Shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Cham, May / June 2022.

[GW18] Ke Gu and Na Wu. Constant size traceable ring signature scheme without random oracles. Cryptology ePrint Archive, Report 2018/288, 2018.

[HC24] Xiangyu Hui and Sid Chi-Kin Chau. tt LLRing: Logarithmic linkable ring signatures with transparent setup. In Joaquin Garcia-Alfaro, Rafał Kozik, Michał Choraś, and Sokratis Katsikas, editors, *ESORICS 2024, Part III*, volume 14984 of *LNCS*, pages 299–319. Springer, Cham, September 2024.

[HKSS22] Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Logarithmic-size (linkable) threshold ring signatures in the plain model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 437–467. Springer, Cham, March 2022.

[KSD+24] Thanh Xuan Khuc, Willy Susilo, Dung Hoang Duong, Fuchun Guo, Kazuhide Fukushima, and Shinsaku Kiyomoto. Traceable ring signatures: Logarithmic-size, without any setup, from standard assumptions. In Joseph K. Liu, Liqun Chen, Shi-Feng Sun, and Xiaoning Liu, editors, *ProvSec 2023, Part I*, volume 14903 of *LNCS*, pages 189–208. Springer, Singapore, September 2024.

[KWT25] George Kadianakis, Benedikt Wagner, and Thomas Thiery. zkfocil: Inclusion list privacy using linkable ring signatures, 2025. Accessed: 2025-09-25.

[LASZ13] Joseph K Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):157–165, 2013.

[LHC23] Wei Liao, Lansheng Han, and Peng Chen. Traceable ring signature scheme in internet of vehicles based on lattice cryptography. In *2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 72–79. IEEE, 2023.

[LHH+23a] Junbin Liang, Jianye Huang, Qiong Huang, Liantao Lan, and Man Ho Allen Au. A lattice-based certificateless traceable ring signature scheme. *Information*, 14(3):160, 2023.

[LHH+23b] Junbin Liang, Qiong Huang, Jianye Huang, Liantao Lan, and Man Ho Allen Au. An identity-based traceable ring signatures based on lattice. *Peer-to-Peer Networking and Applications*, 16(2):1270–1285, 2023.

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Berlin, Heidelberg, May 2016.

[LLNW17] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based PRFs and applications to E-cash. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 304–335. Springer, Cham, December 2017.

[LRR+19] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. Omniring: Scaling private payments without trusted setup. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 31–48. ACM Press, November 2019.

[LW06] Joseph K Liu and Duncan S Wong. Enhanced security models and a generic construction approach for linkable ring signature. *International Journal of Foundations of Computer Science*, 17(06):1403–1422, 2006.

[LWC+19] Wulu Li, Yongcan Wang, Lei Chen, Xin Lai, Xiao Zhang, and Jiajun Xin. Fully auditable privacy-preserving cryptocurrency against malicious auditors. *Cryptology ePrint Archive*, 2019.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04*, volume 3108 of *LNCS*, pages 325–335. Springer, Berlin, Heidelberg, July 2004.

[Noe15] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015.

[NTWZ19] Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New code-based privacy-preserving cryptographic constructions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 25–55. Springer, Cham, December 2019.

[PGLZ21] Xin Peng, Ke Gu, Zhenlin Liu, and Wenbin Zhang. Traceable identity-based ring signature for protecting mobile iot devices. In *International Conference on Data Mining and Big Data*, pages 158–166. Springer, 2021.

[QW22] Yanhong Qi and Li-Ping Wang. A new code-based traceable ring signature scheme. *Security and Communication Networks*, 2022(1):3938321, 2022.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Berlin, Heidelberg, December 2001.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SZ21] Alessandra Scafuro and Bihan Zhang. One-time traceable ring signatures. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021, Part II*, volume 12973 of *LNCS*, pages 481–500. Springer, Cham, October 2021.

[TSS+18] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0). In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 558–576. Springer, Cham, July 2018.

[TW05] Patrick P Tsang and Victor K Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*, pages 48–60. Springer, 2005.

[TWC+04] Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 384–398. Springer, Berlin, Heidelberg, December 2004.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Berlin, Heidelberg, April 2015.

[WLB+23] Wei Wei, Min Luo, Zijian Bao, Cong Peng, and Debiao He. Traceable ring signatures from group actions: Logarithmic, flexible, and quantum resistant. In *SAC Workshop 2023 Preproceedings*, 2023. Preproceedings, SAC Workshop 2023.

[WLB+24] Wei Wei, Min Luo, Zijian Bao, Cong Peng, and Debiao He. Traceable ring signatures from group actions: Logarithmic, flexible, and quantum resistant. In Claude Carlet, Kalikinkar Mandal, and Vincent Rijmen, editors, *SAC 2023*, volume 14201 of *LNCS*, pages 169–188. Springer, Cham, August 2024.

[XLAZ24] Yuxi Xue, Xingye Lu, Man Ho Au, and Chengru Zhang. Efficient linkable ring signatures: New framework and post-quantum instantiations. In Joaquin Garcia-Alfaro, Rafał Kozik, Michał Choraś, and Sokratis Katsikas, editors, *ESORICS 2024, Part IV*, volume 14985 of *LNCS*, pages 435–456. Springer, Cham, September 2024.

[XTA+25] Min Xie, Zhengzhou Tu, Man Ho Au, Junbin Fang, Xuan Wang, and Zoe Lin Jiang. Efficient constant-size linkable ring signatures for ad-hoc rings via pairing-based set membership arguments. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 858–872. ACM Press, October 2025.

[XZC+24] Jinghuan Xie, Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Kim-Kwang Raymond Choo. Linkable, k-times traceable and revocable ring signature for fine-grained accountability in blockchain transactions. *IEEE Internet of Things Journal*, 2024.

[YEL+21] Tsz Hon Yuen, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. DualRing: Generic construction of ring signatures with efficient instantiations. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 251–281, Virtual Event, August 2021. Springer, Cham.

[YLGT23] Qing Ye, Yongkang Lang, Hongfu Guo, and Yongli Tang. Efficient lattice-based traceable ring signature scheme with its application in blockchain. *Information Sciences*, 648:119536, 2023.

[YSL+20] Tsz Hon Yuen, Shifeng Sun, Joseph K. Liu, Man Ho Au, Muhammed F. Esgin, Qingzhao Zhang, and Dawu Gu. RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 464–483. Springer, Cham, February 2020.

[YW25] Luona Yin and Huaqun Wang. Conditional privacy-preserving spectrum trading scheme based on traceable ring signature for dss system. *The Journal of Supercomputing*, 81(1):14, 2025.

[ZCW+24] Xiao Zhao, Suzhen Cao, Zheng Wang, Dandan Xing, and Dawei Zhou. A traceable and anonymous authentication ring signature scheme with privacy protection. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 570–575. IEEE, 2024.

A Hardness Assumptions

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g. We recall the DLog assumption and the DDH assumption here.

Definition 12 (The DLog assumption). *The Discrete Logarithm (DLog) assumption states that, for any PPT adversary $\mathcal{A}$, the adversary $\mathcal{A}$'s advantage*

$$\mathsf{Adv}^{dl}_{\mathbb{G},\mathcal{A}}(\lambda) := \Pr[x \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g, g^x) = x] \leq negl(\lambda).$$

Definition 13 (The DDH assumption). *The Decisional Diffie-Hellman (DDH) assumption states that, for any PPT adversary $\mathcal{A}$, the adversary $\mathcal{A}$'s advantage*

$$\mathsf{Adv}^{ddh}_{\mathbb{G},\mathcal{A}}(\lambda) := |\Pr[x, y \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y, g^z) = 1]| \leq negl(\lambda).$$

It has been proved that the DDH assumption implies the following with a tight reduction [EHK+13, GHKW16], for any $Q = poly(\lambda)$.

$$\mathsf{Adv}^{Q\text{-}ddh}_{\mathbb{G},\mathcal{A}}(\lambda) := \left| \begin{array}{l} \Pr\left[x, \omega_1, ..., \omega_Q \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}\begin{pmatrix} g & g^{\omega_1} & ... & g^{\omega_Q} \\ g^x & g^{\omega_1 x} & ... & g^{\omega_Q x} \end{pmatrix} = 1\right] - \\ \Pr\left[x, \omega_1, ..., \omega_Q, u_1, ..., u_Q \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}\begin{pmatrix} g & g^{\omega_1} & ... & g^{\omega_Q} \\ g^x & g^{u_1} & ... & g^{u_Q} \end{pmatrix} = 1\right] \end{array} \right| \leq negl(\lambda).$$

B Proof of Theorem 2

Proof (of Theorem 2). The first direction is straightforward, since the condition win'_3 in the variant linkability experiment is stricter than the condition win_3 in the (extended) linkability experiment.

Now consider the other direction. At a high level, linkability and variant linkability differ only at conditions win_3 and win'_3, i.e., if the adversary can enlarge its attacking advantage by including uncorrupted users into the ring. Recall that whether two signatures are linked is only determined by the event label e and the corresponding secret keys. If the adversary breaks linkability but not the variant linkability, then it must be the case that the adversary forges a signature that is linked to one extra and uncorrupted user in the ring, which, is infeasible due to the extended exculpability.

More precisely, we prove that if there is an adversary $\mathcal{A}$ that breaks the linkability, then there is a reduction algorithm $\mathcal{A}'$ that either breaks the variant linkability, or the extended exculpability. $\mathcal{A}'$ has access to oracles NewUser, Corr, NewCorr, Sign from its own challenger, and it simulates the linkability experiment for $\mathcal{A}$. Let $(\mathsf{e}^*, \{(R_i, \mathsf{m}_i, \sigma_i)\}_{i\in[\ell+1]})$ be $\mathcal{A}$'s final output, and win_1, win_2, win_3, and win_4 in $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ are satisfied. Here win_3 means that $|\mathcal{S}_{\mathsf{corr}} \cap (\bigcup_{i\in[\ell+1]} R_i)| \leq \ell$. We analyze in two cases.

- **Case 1.** $\mathsf{win}'_3 = |\bigcup_{i\in[\ell+1]} R_i| \leq \ell$ is also satisfied.
  Then $\mathcal{A}'$ just forward $\mathcal{A}$'s outputs to its own challenger, which is a direct attack against the variant linkability.
- **Case 2.** win'_3 is not satisfied.
  Let $\mathcal{S}_h = \bigcup_{i\in[\ell+1]} R_i - \mathcal{S}_{\mathsf{corr}}$. $\mathcal{A}'$ samples $|\mathcal{S}_h|$ fresh messages $\{\mathsf{m}'_j\}_{j\in\mathcal{S}_h}$ that differ from all messages appear in $\mathcal{A}$'s final output and the simulation of Sign up to now. Then $\mathcal{A}'$ queries $\textsc{Sign}(j, \{j\}, \mathsf{e}^*, \mathsf{m}'_j)$ for all $j \in \mathcal{S}_h$ and obtain signatures $\{\sigma'_j\}_{j\in\mathcal{S}_h}$. There are two subcases.
  - **Case 2.1.** There exists $j^*, j \in \mathcal{S}_h$ and $i \in [\ell+1]$ such that $\mathsf{Trace}(\mathsf{e}^*, \sigma'_{j^*}, \sigma_i) = \mathsf{linked}$ or $\mathsf{Trace}(\mathsf{e}^*, \sigma'_j, \sigma_i) = pk_{j^*}$.
    This indicates an attack against the extended exculpability for honest user j^*. The reduction is similar to Case 1 in the proof of Theorem 1.

- **Case 2.2.** For all $j \in \mathcal{S}_h$ and $i \in [\ell+1]$ it holds that $\mathsf{Trace}(\mathsf{e}^*, \sigma'_j, \sigma_i) = \mathsf{indep}$.
  In this case, $\mathcal{A}'$ queries $\textsc{Corr}(j)$ for all $j \in \mathcal{S}_h$ to obtain the secret keys $\{sk_j\}_{j\in\mathcal{S}_h}$. Then, it samples another $|\mathcal{S}_h|$ fresh messages $\{\mathsf{m}''_j\}_{j\in\mathcal{S}_h}$ that differ from all messages appear up to now, and invokes $\mathsf{Sign}(\{pk_j\}, sk_j, \mathsf{e}^*, \mathsf{m}''_j)$ to obtain σ''_j. Finally $\mathcal{A}'$ outputs

$$(\mathsf{e}^*, \{(R_i, \mathsf{m}_i, \sigma_i)\}_{i\in[\ell+1]}, \{(\{j\}, \mathsf{m}''_j, \sigma''_j)\}_{j\in\mathcal{S}_h})$$

  as its final attack against the variant linkability.
  We now analyze that $(\mathsf{win}_1 \wedge \mathsf{win}_2 \wedge \mathsf{win}'_3 \wedge \mathsf{win}_4)$ is satisfied in the experiment $\mathsf{Exp}^{v\text{-}link}_{\mathsf{TRS},\mathcal{A}'}(\lambda)$.
  1. win_1 (all signatures are valid): This is implied by win_1 in $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ and the correctness.
  2. win_2 (all signatures are not related to the signing oracle): This is implied by win_2 in $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$ and the freshness of $\{\mathsf{m}''_j\}$.
  3. win'_3 (the union set of rings is smaller than the number of outputted signatures): This is implied by $|\mathcal{S}_{\mathsf{corr}} \cap (\bigcup_{i\in[\ell+1]} R_i)| \leq \ell$ and $\mathcal{S}_h = \bigcup_{i\in[\ell+1]} R_i - \mathcal{S}_{\mathsf{corr}}$.
  4. win_4 (all signatures are pairwise independent): This is implied by win_4 in $\mathsf{Exp}^{link}_{\mathsf{TRS},\mathcal{A}}(\lambda)$, the condition of Case 2.2., and the correctness (that $\mathsf{Trace}(\mathsf{e}^*, \sigma''_j, \sigma''_k) = \mathsf{indep}$ for honestly generated σ''_j and σ''_k using sk_j and sk_k $(j, k \in \mathcal{S}_h)$).

  Therefore, $\mathcal{A}'$ successfully breaks the variant linkability as defined in Def. 11, concluding the proof. □

C Proof of Theorem 5

Before proving Theorem 5, we first provide the following key representation lemma.

Lemma 1 (Key Representation Lemma). *Let $H' : \{0,1\}^* \to \mathbb{G}$ be modeled as a random oracle, and let $\mathcal{A}$ be a PPT adversary making at most $Q_{H'}$ queries to H' and outputting rings of size at most n. Let ε denote the probability that $\mathcal{A}$ outputs (PK_R, δ, x, e) with $PK_R = (X_1, \ldots, X_n)$, $\delta \in [n]$, $e \in \mathbb{Z}_q \setminus \{0\}$ and*

$$X_\delta \;=\; g^x \cdot H'(PK_R)^e.$$

Then there is a PPT algorithm $\mathcal{B}$ with

$$\frac{\varepsilon^2}{n\,Q_{H'}} - \frac{1}{q} \;\leq\; \mathsf{Adv}^{dlog}_{\mathbb{G},\mathcal{B}}(\lambda).$$

Proof. $\mathcal{B}$ takes as input a DLog instance (g, Y) and outputs $y = \log_g Y$. It answers the j-th fresh query $H'(R_j)$ by sampling $a_j, b_j \xleftarrow{\$} \mathbb{Z}_q$ and returning $\tilde{g}_j := Y^{a_j} g^{b_j}$. Since b_j is uniform, $\tilde{g}_j$ is uniformly distributed over $\mathbb{G}$, so the simulation is perfect and (a_j, b_j) remains hidden from $\mathcal{A}$. All honest key pairs are generated by $\mathcal{B}$ itself, hence every oracle of the experiment is simulated exactly.

First run. $\mathcal{B}$ guesses in advance the index $i^* \in [Q_{H'}]$ of the query on which $\mathcal{A}$ succeeds together with $\delta \in [n]$, which is correct with probability $1/(n\,Q_{H'})$. By setting $\tilde{g} := \tilde{g}_{i^*} = Y^a g^b$, a successful run yields

$$X_\delta \;=\; g^x\,\tilde{g}^{\,e} \;=\; g^{\,x+be}\,Y^{\,ae}. \tag{1}$$

Fork. $\mathcal{B}$ rewinds $\mathcal{A}$ to the point just before the i^*-th query is answered, samples fresh $a', b' \xleftarrow{\$} \mathbb{Z}_q$, replies with $\tilde{g}' := Y^{a'} g^{b'}$, and answers all subsequent queries with fresh randomness. If $\mathcal{A}$ succeeds again on the same query index, then

$$X_\delta \;=\; g^{x'}\,\tilde{g}'^{\,e'} \;=\; g^{\,x'+b'e'}\,Y^{\,a'e'}. \tag{2}$$

Crucially, the i^*-th oracle *input* is the ring PK_R itself and is fixed before the forking point; hence PK_R, and with it X_δ, is identical in both runs.

Dividing (1) by (2) gives

$$Y^{ae-a'e'} = g^{(x'+b'e')-(x+be)},$$

so whenever $ae \neq a'e'$, $\mathcal{B}$ outputs $y = \big((x'+b'e')-(x+be)\big)\cdot(ae-a'e')^{-1}$.

It is left to bound $\Pr[ae = a'e']$. Condition on the entire view of $\mathcal{A}$ in the second run, for every $a' \in \mathbb{Z}_q$ there is a unique b' with $Y^{a'}g^{b'} = \tilde{g}'$, so a' remains uniform over $\mathbb{Z}_q$. As e' is a deterministic function of that view and $e' \neq 0$, we get $\Pr[a' = ae/e'] = 1/q$. The claimed bound now follows from the forking lemma [BN06]. □

Now we prove Theorem 5 (the security of ZKPoK for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$).

Proof (of Theorem 5). The completeness is straightforward. The simulator for the zero-knowledge property is shown in Fig. 6. One can see if the reprogram does not abort, then the simulated proofs have identical distribution with the proofs from Prove. Let Q_H denotes the total number of queries to H^1 and H^2.

- $\mathsf{info}_1 = (ins||(\hat{A}, C, D, C_1, C_2))$, where $C \leftarrow \mathbb{G}$. Therefore, the probability that reprogramming $H^1(\mathsf{info}_1)$ fails is bounded by Q_H/q.
- $\mathsf{info}_2 = (\mathsf{info}_1||A, S, s, s_D))$, where $S \xleftarrow{\$} \mathbb{G}$. Therefore, the probability that reprogramming $H^2(\mathsf{info}_2)$ fails is bounded by Q_H/q.
- $\mathsf{info}_3 = (\mathsf{info}_2||(T_1, T_2))$, where $T_2 \xleftarrow{\$} \mathbb{G}$. Therefore, the probability that reprogramming $H^1(\mathsf{info}_3)$ fails is bounded by Q_H/q.

As a result, the reprogramming fails with probability at most $3Q_H/q$, which concludes the proof of zero-knowledge.

We follow [BBB+18, GOP+22] and rely on the forking lemma [BN06] to prove strong simulation extractability. At a high level, the extractor Ext runs the prover on more than n distinct values of the challenge w and on two distinct values of p, and further invokes the extractor of Π_{IP}, which requires $O(n^2)$ valid proof transcripts. Besides the witness $\mathbf{a}_L$, $\mathbf{a}_R$ and x, we also recover the commitment randomness r_A and r_C; both are needed in the last step below, where the two openings of C are related to each other. Rewinding on w and p suffices for this purpose, and in total Ext invokes the prover $O(n^3)$ times.

To facilitate a better understanding of the proof structure, we list all elements that appear in accordance with the signing procedure as follows.

crs:	$\mathbf{p}, \mathbf{v}, \tilde{g} = H'(X_1, ..., X_n)$
instance:	$X_1, ..., X_n, g_{\mathsf{e},1}, g_{\mathsf{e},2}, \mathsf{m}, pid_1, pid_2$
witness:	$\mathbf{a}_L, \mathbf{a}_R, x$ // $\mathbf{a}_L$ the binary vector of δ
	$\hat{A}, C, D, C_1, C_2$
w	
	$A, S, s, s_D, \mathbf{u}$
y, z	
	$T_1, T_2, \mathbf{v}'$
p	
	$r_{\hat{t}}, r', \hat{t}, \boxed{\mathbf{l}, \mathbf{r}}$ // $\mathbf{l}, \mathbf{r}$ extracted from Π_{IP}

At first, Ext utilizes the extractor of Π_{IP} to obtain witness $\mathbf{l}, \mathbf{r}$ such that $A' = \mathbf{u}^{\mathbf{l}}(\mathbf{v}')^{\mathbf{r}}$ and $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$. Here $A' = A \cdot S^p \cdot \mathbf{u}^{-z}(\mathbf{v}')^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{1}} \cdot \tilde{g}^{-r'}$. By presenting $A = \mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}$, $S = \mathbf{u}^{\mathbf{s}_L}\mathbf{v}^{\mathbf{s}_R}\tilde{g}^{r_S}$, and $\mathbf{v}' = \mathbf{v}^{y^{-n}}$, we get

$$\mathbf{u}^{\mathbf{l}}(\mathbf{v}')^{\mathbf{r}} = (\mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_A}) \cdot (\mathbf{u}^{\mathbf{a}_L}\mathbf{v}^{\mathbf{a}_R}\tilde{g}^{r_S})^p \cdot \mathbf{u}^{-z}(\mathbf{v}')^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{1}} \cdot \tilde{g}^{-r'},$$

which can be rewritten as

$$\mathbf{u}^{\mathbf{l}} \cdot \mathbf{v}^{(\mathbf{y}^{-n}\circ\mathbf{r})} \cdot \tilde{g}^{r'} = \mathbf{u}^{(\mathbf{a}_L + \mathbf{s}_L\cdot p - z\cdot\mathbf{1})} \cdot \mathbf{v}^{(\mathbf{a}_R + \mathbf{s}_R\cdot p + z\cdot\mathbf{1} + z^2\cdot\mathbf{y}^{-n})} \cdot \tilde{g}^{r_A + r_S\cdot p}.$$

Recall that $\mathbf{u}, \mathbf{v}, \tilde{g}$ are all random elements over $\mathbb{G}$. Under the DLog assumption, we obtain

$$\mathbf{l} = \mathbf{a}_L + \mathbf{s}_L \cdot p - z \cdot \mathbf{1}$$
$$\mathbf{y}^{-n} \circ \mathbf{r} = \mathbf{a}_R + \mathbf{s}_R \cdot p + z \cdot \mathbf{1} + z^2 \cdot \mathbf{y}^{-n}$$
$$r' = r_A + r_S \cdot p$$

Therefore, collecting two proofs with distinct p and $\mathbf{l}, \mathbf{r}$, we are able to recover $\mathbf{a}_L, \mathbf{a}_R, \mathbf{s}_L, \mathbf{s}_R$. On the other hand, if the above equations do not hold for both challenges and $\mathbf{l}, \mathbf{r}$ from the proofs, then we have two distinct representations of the same group element using a set of independent generators, which breaks the DLog assumption.

To extract witness x, we present $C = g^x \tilde{g}^{r_C}$ and $D = C^w \cdot g^s \tilde{g}^{s_D}$. Then we get

$$D = (g^x \tilde{g}^{r_C})^w g^s \tilde{g}^{s_D} = g^{(w \cdot x + s)} \tilde{g}^{(w \cdot r_C + s_D)}.$$

From two proof transcripts with the same C, D but different w and s, s_D, we can recover x and r_C. On the other hand, if the above equations do not hold for both proof transcripts, then we have two distinct representations of the same group element from g and $\tilde{g}$, which breaks the DLog assumption.

Before completing the proof, we fix how the two rewinding layers compose. We regard $\mathsf{Ext}^{\mathcal{A}}$, the extractor together with the prover it runs, as a single algorithm $\mathcal{A}'$ that in one execution outputs the tuple required by Lemma 1. Extraction is carried out first, within a single execution of the random oracle H', and Lemma 1 then forks $\mathcal{A}'$ at its i^*-th query to H'. Only H' is re-answered on this fork, so $\tilde{g}$ is constant throughout any single execution of Ext and all transcripts it collects refer to the same commitment generator. We let $Q_{H'}$ bound the total number of queries to H' made by $\mathcal{A}'$ across all its rewindings, and note that the bound of Lemma 1 is quadratic in the success probability of $\mathcal{A}'$, which already absorbs the losses of the inner extraction.

It remains to show that the extracted $\mathbf{a}_L$ and x are consistent, namely $X_\delta = g^x$ for the index δ with $a_{L,\delta} = 1$. Recall that $\hat{A}$ and C are sent before the challenge w is sampled, and that the verifier recomputes $A := C^w \hat{A}$. For each challenge w on which extraction succeeds, the argument above yields a vector $\mathbf{a}_L^{(w)} \in \{0,1\}^n$ of Hamming weight one, together with $\mathbf{a}_R^{(w)} = \mathbf{a}_L^{(w)} - \mathbf{1}$ and a randomness $r_A^{(w)}$, such that $A = \mathbf{u}^{\mathbf{a}_L^{(w)}} \mathbf{v}^{\mathbf{a}_R^{(w)}} \tilde{g}^{r_A^{(w)}}$ with $\mathbf{u} = \mathbf{pk}^w \circ \mathbf{p}$. Substituting $\mathbf{u}$ and $A = C^w \hat{A}$, we get

$$C^w \, \hat{A} \;=\; \left(X_{\delta(w)}\right)^w \cdot \mathbf{p}^{\mathbf{a}_L^{(w)}} \mathbf{v}^{\mathbf{a}_R^{(w)}} \tilde{g}^{r_A^{(w)}}, \tag{3}$$

where $\delta(w) \in [n]$ is the unique index with $a_{L,\delta(w)}^{(w)} = 1$, so that $\mathbf{pk}^{\mathbf{a}_L^{(w)}} = X_{\delta(w)}$.

Note that (3) is stated with respect to the challenge-dependent basis $\mathbf{u}$, so a priori the extracted vectors may differ from branch to branch. We rule this out by a counting argument. Since Ext collects more than n distinct challenges w, which is within the transcript budget already required by Π_{IP}, while $\delta(w)$ ranges over $[n]$, there are two distinct challenges $w \neq \bar{w}$ with $\delta(w) = \delta(\bar{w}) =: \delta$. For this pair the extracted vectors coincide, $\mathbf{a}_L^{(w)} = \mathbf{a}_L^{(\bar{w})} =: \mathbf{a}_L$ and $\mathbf{a}_R^{(w)} = \mathbf{a}_R^{(\bar{w})} = \mathbf{a}_L - \mathbf{1}$, since a binary vector of Hamming weight one is determined by its support. Writing $r_A := r_A^{(w)}$ and $\bar{r}_A := r_A^{(\bar{w})}$ and dividing the two instances of (3), both $\hat{A}$ and the term $\mathbf{p}^{\mathbf{a}_L} \mathbf{v}^{\mathbf{a}_R}$ cancel, leaving

$$C^{w-\bar{w}} = X_\delta^{w-\bar{w}} \cdot \tilde{g}^{r_A - \bar{r}_A}, \;\text{ hence } C = X_\delta \cdot \tilde{g}^{\rho}, \;\; \rho := (r_A - \bar{r}_A)\,(w - \bar{w})^{-1}.$$

On the other hand, the extraction of x above gives $C = g^x \tilde{g}^{r_C}$. Combining the two openings of C yields $X_\delta = g^x \cdot \tilde{g}^{r_C - \rho}$. If $r_C \neq \rho$, then $(PK_R, \delta, x, r_C - \rho)$ is precisely an output of the form excluded by Lemma 1, and we obtain a PPT algorithm solving DLog. Therefore $r_C = \rho$ and $X_\delta = g^x$, so (δ, x) is a valid witness for $\mathcal{L}_{\mathsf{DL\text{-}TRS}}$, which concludes the proof of strong simulation extractability. □

D Flaws of the Security Proof for Wei et al.'s Scheme [WLB+24, WLB+23]

Wei et al. [WLB+24, WLB+23] proposed a general TRS scheme based on restricted pair of group actions, OR sigma protocol and collision-resistant hash functions, and gave two instantiations from isogenies and lattices, respectively. We recall the main result of Wei et al.'s schemes here:

[WLB+23, Thm. 1, p.14]: If the OR sigma protocol is soundness and zero-knowledge, the hash function H_1, H_2 are collision-resistant, the ResPGA is a restricted pair of group actions, then our TRS scheme Π_{ISO} satisfies tag-linkability, anonymity and exculpability.

Here we point out two flaws in [WLB+23]'s analysis.

Flaw 1. The insufficiency of collision-resistance for hash functions.

In the analysis of (trace) correctness [WLB+23, Situation 2, p.14], the authors claimed that "it is hard to find i such that $H_1(a,i) - H_1(a',i) = H_1(a,\pi) - H_1(a',\pi)$ since the collision resistance of H_1". Here π denotes the index of an honest signer and $i \neq \pi$, $a \neq a'$. However, the above claim cannot be guaranteed due to the collision resistance of H_1. To see this, consider the following group-based hash functions:

$$H_1(\alpha, \beta) = g^{\alpha} \cdot h^{\beta}.$$

The collision resistance of H_1 relies on the DLog assumption. However, it is easy to see that for any i, π, a, a', we have

$$H_1(a,i) - H_1(a',i) = g^{a-a'} = H_1(a,\pi) - H_1(a',\pi).$$

In fact, any H_1 of the form

$$H_1(\alpha, \beta) = H_2(\alpha) + H_3(\beta)$$

is a counterexample, even if H_2 and H_3 are modeled as random oracle.

Flaw 2. The insufficiency of soundness.

Though there is no formal definition of soundness in [WLB+23], we still believe that the soundness of the Sigma protocol (or, the resulted non-interactive zero-knowledge proofs) is insufficient. The following arguments rely on the fact that we can extract a secret key (a witness) from the adversary's final output. Therefore, the stronger notion of simulation extractability seems to be mandatory to complete the proof.

E Insecurity of Qi and Wang's TRS Scheme [QW22]

Qi and Wang [QW22] proposed a code-based TRS scheme, whose security is based on the hardness of the syndrome decoding problem and 2-regular null syndrome decoding problem. However, we point out that the scheme does not achieve anonymity.

Let $\mathbf{s}_i^{\top} = \mathbf{H}\mathbf{e}_i^{\top}$ be the public key of the i-th user with $\mathbf{e}_i$ the corresponding secret key. We recall the signing algorithm [QW22, Algorithm 3] here (we omit the descriptions of the data structure since they are irrelevant with the insecurity discussed here):

To sign a message m, the user i does:

1. compute $\tilde{H} = H_1(\mathsf{e})$ and $\tilde{H}\mathbf{e}_i^{\top} = \mathbf{r}_i^{\top}$.
2. compute the sequence $r_1, ..., r_{i-1}, r_{i+1}., ..., r_n$ according to some recurrence relation determined by the message m.
3. compute $\mathbf{d}_j = H_2(\mathbf{s}_j, \mathbf{r}_j)$ for all $j \in [n]$.
4. accumulate $\{\mathbf{d}_j\}_{j\in[n]}$ as $\mathbf{u}$ and generates a witness $\mathbf{w}_i$ proving that $\mathbf{d}_i$ was accumulated into $\mathbf{u}$.
5. generate a zero-knowledge proof π for instance $X = (\mathbf{H}, \tilde{H}, \mathbf{s}_i, \mathbf{r}_i, \mathbf{B}, \mathbf{d}_i, \mathbf{u})$ and witness $W = (\mathbf{w}_i, \mathbf{e}_i)$.
6. Return $\sigma = (r_1, ..., r_n, \pi)$.

As observed, the signer's corresponding values $\mathbf{s}_i, \mathbf{r}_i, \mathbf{d}_i$ are explicitly revealed in the instance, thereby exposing the signer's identity. Though [QW22] proposed an accumulated-GStern's protocol and applied the accumulator in [NTWZ19] to hide secret key information, it did not provide any 1-out-of-n membership proof, which is essential for all ring signatures.