Group Signatures and More from Isogenies and Lattices: Generic, Simple, and Efficient

Ward Beullens[1], Samuel Dobson[2], Shuichi Katsumata[3], Yi-Fu Lai[2], Federico Pintore[4]

[1]imec-COSIC, KU Leuven, Belgium
`ward.beullens@esat.kuleuven.be`
[2]University of Auckland, New Zealand
`samuel.dobson.nz@gmail.com` `ylai276@aucklanduni.ac.nz`
[3]National Institute of Advanced Industrial Science and Technology (AIST), Japan
`shuichi.katsumata@aist.go.jp`
[4]Department of Mathematics, University of Bari, Italy
`federico.pintore@uniba.it`

25th October 2021

Abstract

We construct an efficient dynamic group signature (or more generally an accountable ring signature) from isogeny and lattice assumptions. Our group signature is based on a simple generic construction that can be instantiated by cryptographically hard group actions such as the CSIDH group action or an MLWE-based group action. The signature is of size $O(\log N)$, where N is the number of users in the group. Our idea builds on the recent efficient OR-proof by Beullens, Katsumata, and Pintore (Asiacrypt'20), where we efficiently add a proof of valid ciphertext to their OR-proof and further show that the resulting non-interactive zero-knowledge proof system is *online extractable*.

Our group signatures satisfy more ideal security properties compared to previously known constructions, while simultaneously having an attractive signature size. The signature size of our isogeny-based construction is an order of magnitude smaller than all previously known post-quantum group signatures (e.g., 6.6 KB for 64 members). In comparison, our lattice-based construction has a larger signature size (e.g., either 126 KB or 89 KB for 64 members depending on the satisfied security property). However, since the $O(\cdot)$-notation hides a very small constant factor, it remains small even for very large group sizes, say 2^{20}.

1 Introduction

Group signature schemes, introduced by Chaum and van Heyst [Cv91], allow authorized members of a group to individually sign on behalf of the group while the specific identity of the signer remains anonymous. However, should the need arise, a special entity called the group manager (or sometimes the tracing authority) can trace the signature to the signer, thus holding the group members accountable for their signatures. Group signatures have been an active area of academic research for the past three decades, and have also been gathering practical attention due to the recent real-world deployment of variants of group signatures such as directed anonymous attestation (DAA) [BCC04] and enhanced privacy ID (EPID) [BL07].

Currently, there are versatile constructions of *efficient* group signatures from *classical* assumptions, e.g., [BBS04, DP06, Gro07, FI06, BCN$^+$10, LPY15, LMPY16, DS18, BHSB19, CS20]. In this work, when we argue the efficiency of a group signature, we focus on one of the quintessential metrics: the signature size. We require it to be smaller than $c \cdot \log N$ bits, where N is the group size and c is some explicit small polynomial in the security parameter. In their seminal work, Bellare, Micciancio, and Warinschi [BMW03] provided a generic construction of a group signature with signature size $O(1)$ from any signature scheme, public-key encryption scheme, and general non-interactive zero-knowledge (NIZK) proof system. Unfortunately, this only provides an asymptotic feasibility result, and thus one of the main focuses of subsequent works, including ours, has been to construct a concretely efficient group signature.

In contrast to the classical setting, constructing efficient group signatures from any *post-quantum* assumptions has been elusive. Since the first lattice-based construction by Gordon, Katz, and Vaikuntanathan [GKV10], there has been a rich line of subsequent works on lattice-based (and one code-based) group signatures, including but not limited to [LLLS13, ELL$^+$15, LLNW16, LNWX18, KY19]. However, these results remained purely asymptotic. It was not until recently that efficient lattice-based group signatures appeared [BCN18, dLS18, EZS$^+$19, ESZ22]. In [ESZ22], Esgin et al. report a signature size of 12KB and 19KB for a group size of $N = 2^6$ and 2^{10}, respectively—several orders of magnitude better than prior constructions.[1] These rapid improvements in efficiency for lattices originate in the recent progress of lattice-based NIZK proof systems for useful languages [YAZ$^+$19, BLS19, ESLL19, ALS20, ENS20, LNS20, LNS21], most of which rely heavily on the properties of special structured lattices. Thus, it seems impossible to import similar techniques to other post-quantum assumptions or to standard non-structured lattices. For instance, constructing efficient group signatures from isogenies—one of the promising alternative post-quantum tools to lattices—still seems out of reach using current techniques. This brings us to the main question of this work:

> *Can we construct an efficient group signature secure from isogenies? Moreover, can we have a generic construction that can be instantiated from versatile assumptions, including those based on less structured lattices?*

In addition, as we discuss in more detail later, we notice that all works regarding efficient post-quantum group signatures [BCN18, KKW18, dLS18, EZS$^+$19, ESZ22] do not satisfy the ideal security properties (which are by now considered standard) formalized by Bootle et al. [BCC$^+$16]. Thus, we are also interested in the following question:

> *Can we construct efficient post-quantum group signatures satisfying the ideal security properties formalized by Bootle et al. [BCC$^+$16]?*

To address these questions, in this work we focus on *accountable ring signatures* [XY04]. An accountable ring signature offers the flexibility of choosing the group of users when creating a signature (like a ring signature [RST01]), while also enforcing accountability by including one of the openers in the group (like a group signature). Although research on accountable ring signatures is still limited [XY04, BCC$^+$15, LZCS16, KP17, EZS$^+$19], we advocate that they are as relevant and interesting as group and ring signatures. As shown by Bootle et al. [BCC$^+$15], accountable ring signatures imply group and ring signatures by naturally limiting or downgrading their functionality. Thus, an efficient post-quantum solution to an accountable ring signature implies solutions for *both* secure (dynamic) group signatures [BSZ05] and ring signatures, making it an attractive target to focus on.

Finally, as an independent interest, we are also concerned with *tightly*-secure constructions. To the best of our knowledge, all prior efficient post-quantum secure group and ring signatures are in the random oracle model and have a very loose reduction loss. In typical security proofs, given an adversary with advantage ϵ that breaks some security property of the group signature, we can only construct an adversary with advantage at most $(N^2 Q)^{-1} \cdot \epsilon^2$ against the underlying hard problem, where Q is the number of random oracle queries and N is the number of users in the system. If we aim for 128-bit security (i.e., $\epsilon = 2^{-128}$), and set for example $(N, Q) = (2^{10}, 2^{50})$, then we need at least 326-bits of security for the hard problem. When aiming for a provably-secure construction, the parameters must be set much larger to compensate for this significant reduction loss, which then leads to a less efficient scheme. This is especially unattractive in the isogeny

---

[1] We note that their signature size grows by $\log^t N$ for a small constant $t > 1$ rather than simply by $\log N$.

setting since only the smallest among the CSIDH parameters [CLM+18] enjoys properties suitable to achieve concrete efficiency [BKV19].

1.1 Our Contribution

In this work, we construct an efficient accountable ring signature based on isogenies and lattices. This in particular implies the first efficient isogeny-based group signature. Our generic construction departs from known general feasibility results such as [BMW03] and builds on primitives that can be efficiently instantiated. Unlike previous efficient post-quantum group signatures, our scheme satisfies all the desired properties provided by Bootle et al. [BCC+16] including *dynamicity* and *fully (CCA) anonymity*: the former states that the group members can be added and revoked dynamically and are not fixed on setup; the later states that anonymity holds even in the presence of an adversary that sees the signing keys of all honest users, who is additionally granted access to an opening oracle. We also satisfy the ideal variant of *non-frameability* and *traceability* [BCC+16], where the former is captured by *unforgeability* in the context of accountable ring signature. Roughly, this ensures that arbitrary collusion among members, even with the help of a corrupted group manager, cannot falsely open a signature to an honest user.

Our accountable ring signature schemes are realized in three steps. We first provide a generic construction of an accountable ring signature from simple cryptographic primitives such as a public-key encryption (PKE) scheme and an accompanying NIZK for a specific language. We then show an efficient instantiation of these primitives based on a group action that satisfies certain cryptographic properties. Finally, we instantiate the group action by either the CSIDH group action or the MLWE-based group action. Our generic construction builds on the recent efficient OR-proofs for isogeny and lattice-based hard languages by Beullens, Katsumata, and Pintore [BKP20], which were used to construct ring signatures. The most technical part of this work is to efficiently add a proof of valid ciphertext to their OR-proof and proving full anonymity, which done naively would incur an exponential security loss. At the core of our construction is an efficient *online-extractable* OR-proof that allows to also prove validity of a ciphertext.

Moreover, thanks to the online extractability, our construction achieves a much tighter reduction loss compared to prior accountable ring signatures (and also group and ring signatures). It suffices to assume that the underlying post-quantum hard problem cannot be solved with advantage more than $N^{-1} \cdot \epsilon$ rather than $(N^2 Q)^{-1} \cdot \epsilon^2$ as in prior works whose proofs rely on the forking lemma [FS87, PS00]. Working with the above example, we only lose 10-bits rather than 198-bits of security. We further show how to remove N^{-1} using the Katz-Wang technique [KW03] along with some techniques unique to our NIZK. As a side product, we obtain a tightly-secure and efficient isogeny and lattice-based ring signatures, improving upon those by Beullens et al. [BKP20] which have a loose security reduction.

Comparison to Prior Work. To the best of our knowledge, Esgin et al. [EZS+19, ESZ22] are the only other work that (implicitly) provide an efficient post-quantum accountable ring signature.[2] Since the efficiency of an accountable ring signature is equivalent to those of the group signature obtained through limiting the functionality of the accountable ring signature, we chose to compare the efficiency of our scheme with other state-of-the-art post-quantum group signatures. Tab. 1 includes a comparison of the signature size and the different notions of security it satisfies. The first two schemes satisfy all the desired security properties of a dynamic group signature formalized by Bootle et al. [BCC+16]. Our scheme is the only one to achieve full CCA anonymity. Esgin et al. [ESZ22] achieves full CPA anonymity, where anonymity is broken once an adversary is given access to an opening oracle; in practice, this means that if a specific signature is once opened to some user, then any signature ever signed by that particular user will loose anonymity. Here, "full" means that the signing key of all the users may be exposed to the adversary. In contrast, Katz, Kolesnikov, and Wang [KKW18] satisfies *selfless* CCA anonymity. While their scheme supports opening oracles, anonymity no longer holds if the signing key used to sign the signature is exposed to the adversary. Moreover, our scheme is the only one that also achieves the ideal variant of non-frameability and traceability [BSZ05, BCC+16] (illustrated in the "Manager Accountability" column). The two schemes [KKW18, ESZ22] assume the group manager honestly executes the opening algorithm and that everyone trusts the output. Put differently, a

---

[2] To be precise, they consider a weaker variant of standard accountable ring signature where no Judge algorithm is considered.

	N					Hardness Assumption	Security Level	Anonymity	Manager Accountable
	2	2^5	2^6	2^{10}	2^{21}				
Isogeny	3.6	6.0	6.6	9.0	15.5	CSIDH-512	$*$	CCA	Yes
Lattice	124	126	126	129	134	MSIS/MLWE	NIST 2	CCA	Yes
Lattice	86	88	89	91	96	MSIS/MLWE	NIST 2	CCA	No
[ESZ22]	/	12	/	19	/	MSIS/MLWE	NIST 2	CPA	No
[KKW18]	/	/	280	418	/	LowMC	NIST 5	selfless-CCA	No

Table 1: Comparison of the signature size (KB) of some concretely efficient post-quantum group signature schemes. The first three rows are our scheme. Manager accountability states whether the (possibly malicious) group manager is accountable when opening a signature to some user. Namely, it is "Yes" when even a malicious group manager cannot falsely accuse an honest user of signing a signature that it hasn't signed.

* 128 bits of classical security and 60 bits of quantum security [Pei20].

malicious group manager can frame any honest members in the group by simply replacing the output of the opening algorithm. In contrast, our scheme remains secure even against malicious group managers since the validity of the output of the opening algorithm is verifiable. That is, even the group manager is held *accountable* in our group signature.

Not only our group signatures satisfy more ideal security properties compared to previous constructions, Tab. 1 shows that our signature size remains competitive. Our isogeny-based group signature based on CSIDH provides the smallest signature size among all post-quantum group signatures, which is $0.6 \log_2(N) + 3$ KB. In contrast, our lattice signature is larger; the scheme in the second (resp. third) row has signature size $0.5 \log_2(N) + 123.5$ KB (resp. $0.5 \log_2(N) + 85.9$ KB). It is smaller compared to [KKW18], while larger compared to [ESZ22]. Compared to the two constructions, our signature size grows much slower with the group size N (see also Footnote 1) and also satisfies stronger security. We thus leave it as an interesting open problem to lower the constants in our construction.

1.2 Technical overview

An accountable ring signature is like a standard ring signature where the ring R also includes an arbitrary opener public key opk of the signer's choice when creating a signature σ. The signature σ remains anonymous for anybody who does not know the corresponding opener secret key osk, while the designated opener can use osk to trace the user who created σ. A ring signature can be thought of as an accountable ring signature where $\mathsf{opk} = \bot$, while a group signature can be thought as an accountable ring signature where there is only a single opener.

General Approach. Our generic construction of an accountable ring signature follows the well-known template of the encrypt-then-prove approach to construct a group signature [Cam97]. The high-level idea is simple. The signer encrypts its verification key vk (or another unique identifier) using the opener's public key opk for a PKE scheme and provides a NIZK proof for the following three facts: the ciphertext ct encrypts vk via opk; vk is included in the ring R; and that it knows a secret key sk corresponding to vk. To trace the signer, the opener simply decrypts ct to recover vk. Notice that the NIZK proof implicitly defines a verifiable encryption scheme [CD00, CS03] since it is proving that ct is a valid encryption for some message vk in R. Below, although our construction can be based on any cryptographically-hard group action, we mainly focus on isogenies for simplicity.

One of the difficulties in instantiating this template using isogeny-based cryptography is that we do not have an efficient verifiable encryption scheme for an appropriate PKE scheme. To achieve full anonymity, most of the efficient group signatures, e.g., [DP06, Gro07, FI06, LPY15, LMPY16, dLS18], use an $\mathsf{IND}\text{-}\mathsf{CCA}$ secure PKE as a building block and construct an efficient NIZK that proves validity of the ciphertext. Full anonymity stipulates that an adversary cannot de-anonymize a signature even if it is provided with an opening oracle, which traces the signatures submitted by the adversary. Roughly, by using an $\mathsf{IND}\text{-}\mathsf{CCA}$ secure PKE as a building block, the reduction can simulate the opening oracle by using the decapsulation oracle provided by the $\mathsf{IND\text{-}CCA}$ game, rather than the opener's secret key. In the classical setting, constructing such an efficient $\mathsf{IND\text{-}CCA}$ secure verifiable encryption scheme is possible using the Cramer-Shoup PKE [CS98] that offers a rich algebraic structure. Unfortunately, in the isogeny setting, although we know how to construct an $\mathsf{IND\text{-}CCA}$ secure PKE based on the Fujisaki-Okamoto transform [FO99], it seems quite difficult to provide an accompanying verifiable encryption scheme as the construction internally uses a hash function modeled as a random oracle. Another approach is to rely on the weaker $\mathsf{IND\text{-}CPA}$ secure PKE but to use a stronger NIZK satisfying *online-extractability* [Fis05]. At a high level, the reduction can use the online-extractor to extract the witness in the ciphertext ct instead of relying on the decapsulation oracle.[3] However, it turns out that even this approach is still non-trivial since we do not have any efficient verifiable encryption scheme for existing isogeny-based PKEs, let alone an accompanying online-extractable NIZK. For instance, most isogeny-based $\mathsf{IND\text{-}CPA}$ secure PKEs are based on the *hashed* version of ElGamal, and to the best of our knowledge, there are no efficient verifiable encryption schemes for hashed ElGamal.

Verifiable Encryption Scheme for a Limited Class of PKE. In this work, we observe that in the context of accountable ring signatures and group signatures, we do not require the full decryption capability of a standard PKE. Observe that decryption is only used by the opener and that it *knows* the ciphertext ct must be an encryption of one of the verification keys included in the ring (or group) R. Therefore, given a ciphertext ct, we only require a mechanism to check if ct encrypts a particular message M, rather than being able to decrypt an arbitrary unknown message. Specifically, the opener can simply run through all the verification keys $\mathsf{vk} \in \mathsf{R}$ to figure out which vk was encrypted in ct. This allows us to use a simple $\mathsf{IND\text{-}CPA}$ secure PKE with limited decryption capability based on the CSIDH group action: Let $E_0 \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ be a fixed and public elliptic curve. The public key is $\mathsf{pk} = (E_0, E := s \star E_0)$, where $\mathsf{sk} = s$ is sampled uniformly at random from the class group $\mathcal{C}\ell(\mathcal{O})$. To encrypt a message $\mathsf{M} \in \mathcal{C}\ell(\mathcal{O})$, we sample $r \leftarrow \mathcal{C}\ell(\mathcal{O})$ and set $\mathsf{ct} = (\mathsf{ct}_0 := r \star E_0, \mathsf{ct}_1 := \mathsf{M} \star (r \star E))$. To check if ct decrypts to M', we check whether ct_1 is equal to $\mathsf{M}' \star (\mathsf{sk} \star \mathsf{ct}_0)$. Note that in general we cannot decrypt when M is unknown since we cannot cancel out $\mathsf{sk} \star \mathsf{ct}_0$ from ct_1. Now, observe that proving ct encrypts $\mathsf{M} \in \mathcal{C}\ell(\mathcal{O})$ is easy since there is a simple sigma protocol for the Diffie-Hellman-like statement $(\mathsf{ct}_0, (-\mathsf{M}) \star \mathsf{ct}_1) = (r \star E_0, r \star E)$, where r is the witness, e.g., [EKP20]. Although this comes closer to what we want, this simple sigma protocol is not yet sufficient since the prover must reveal the message M to run it. Specifically, it proves that ct is an encryption of M, while what we want to prove is that ct is an encryption of *some* $\mathsf{M} \in \mathsf{R}$. In the context of accountable ring signature and group signature, this amounts to the signer being able to hide its verification key $\mathsf{vk} \in \mathsf{R}$.

Constructing NIZK *for Accountable Ring Signature.* Let us move forward to the intermediate goal of constructing a (non-online-extractable) NIZK proof system for the following three facts: the ciphertext ct encrypts vk via pk; vk is included in the ring R; and that the prover knows a secret key sk corresponding to vk. Recently, Beullens, Katsumata, and Pintore [BKP20] proposed an efficient sigma protocol (and a non-online-extractable NIZK via the Fiat-Shamir transform) for proving the last two facts, which in particular constitutes an efficient OR-proof. We show how to glue the above "weak" verifiable encryption scheme with their OR-proof.

We first review a variant of the OR-sigma protocol in [BKP20] with proof size $O(N)$, where N is the size of the ring. Assume each user $i \in [N]$ in the ring holds $\mathsf{vk}_i = (E_0, E_i := s_i \star E_0) \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)^2$ and $\mathsf{sk}_i = s_i \in \mathcal{C}\ell(\mathcal{O})$. To prove $\mathsf{vk}_I \in \mathsf{R}$ and that it knows sk_I, the prover first sample $s' \leftarrow \mathcal{C}\ell(\mathcal{O})$ and sets $R_i = s' \star E_i$ for $i \in [N]$. It also samples randomness rand_i and creates commitments $(\mathsf{C}_i = \mathsf{Com}(R_i, \mathsf{rand}_i))_{i \in [N]}$, where this commitment is simply instantiated by a random oracle. It finally samples a random permutation ϕ over $[N]$ and sends a permuted tuple $(\mathsf{C}_{\phi(i)} = \mathsf{Com}(R_i, \mathsf{rand}_i))_{i \in [N]}$. The verifier samples a random bit $b \in \{0, 1\}$. If $b = 0$, the prover returns all the randomness $(s', (\mathsf{rand}_i)_{i \in [N]}, \phi)$ used to create the first message. The verifier then checks if the first message sent by the prover is consistent with this randomness. Otherwise, if $b = 1$, the prover returns $(I'', \mathsf{rand}'', s'') := (\phi(I), \mathsf{rand}_I, s' + s_I)$. The verifier then checks if $\mathsf{C}_{I''} = \mathsf{Com}(s'' \star E_0, \mathsf{rand}'')$ holds. Notice that if the prover is honest, then $s'' \star E_0 = s' \star E_I$ as desired. It is easy to check it is honest-verifier zero-knowledge. The transcript when $b = 0$ is independent of the witness, while the transcript when $b = 1$ can be simulated if the commitment scheme is hiding. Moreover, special soundness can be checked by

---

[3] Note that extractability via rewinding is insufficient for full anonymity as it will cause an exponential reduction loss when trying to extract the witness from adaptively chosen signatures [BFW15].

noticing that given s'' and s', we can extract some (i^*, s^*) such that $(E_0, E_{i^*} = s^* \star E_0) \in \mathsf{R}$. A full-fledged OR-sigma protocol with proof size $O(N)$ is then obtained by running this protocol λ-times in parallel, where λ denotes the security parameter. [BKP20] showed several simple optimization techniques to compress the proof size from $O(N)$ to $O(\log N)$, but we first explain our main idea below.

We add our "weakly decryptable" PKE to this OR-sigma protocol. Since our PKE only handles messages in $\mathcal{C}\ell(\mathcal{O})$, the prover with $\mathsf{vk}_I \in \mathsf{R}$ encrypts the index $I \in [N]$ rather than vk_I, where we assume the verification keys in the ring R are ordered lexicographically.[4] The statement now consists of the ring R and the ciphertext $\mathsf{ct} = (\mathsf{ct}_0 := r \star E_0, \mathsf{ct}_1 = I \star (r \star E))$, where (E_0, E) is the opener's public key opk. Recall the opener can decrypt ct with knowledge of the ring R by brute-force searching for an $i \in [N]$ such that $\mathsf{ct}_1 = i \star (\mathsf{osk} \star \mathsf{ct}_0)$. Now, to prove vk_I is an entry in R and that it knows sk_I, the prover samples $s' \leftarrow \mathcal{C}\ell(\mathcal{O})$ and sets $R_i = s' \star E_i$ for $i \in [N]$ as before. It then further samples $r' \leftarrow \mathcal{C}\ell(\mathcal{O})$ and prepares $\mathsf{ct}'_i = (r' \star \mathsf{ct}_0, (-i) \star (r' \star \mathsf{ct}_1))$ for all $i \in [N]$. Observe that ct'_i is an encryption of the message $(I - i)$ using randomness $(r' + r)$. Specifically, ct'_I is of the form $((r' + r) \star E_0, (r' + r) \star E)$, which admits a natural sigma protocol as explained above. Finally, the prover samples randomness rand_i and a random permutation ϕ over $[N]$, and sends the randomly permuted commitments $(\mathsf{C}_{\phi(i)} = \mathsf{Com}(R_i \| \mathsf{ct}'_i, \mathsf{rand}_i))_{i \in [N]}$. The verifier samples a random bit $b \in \{0, 1\}$. If $b = 0$, then similarly to the above OR-sigma protocol, the prover simply returns all the randomness and the verifier checks the consistency of the first message. Otherwise, if $b = 1$, the prover returns $(I'', \mathsf{rand}'', s'', r'') := (\phi(I), \mathsf{rand}_I, s' + s_I, r' + r)$. The verifier checks if $\mathsf{C}_{I''} = \mathsf{Com}(s'' \star E_0 \| (r'' \star E_0, r'' \star E), \mathsf{rand}'')$ holds. Correctness and honest-verifier zero-knowledge holds essentially for the same reason as the above OR-sigma protocol. More importantly, special soundness holds as well. Intuitively, since the opening to $b = 0$ forces the cheating prover to commit to the proper (vk_i, i)-pair, a cheating prover cannot encrypt an index I' and prove that it has sk_I corresponding to vk_I for a different $I \neq I'$.

To compile our sigma protocol into an NIZK, we apply the Fiat-Shamir transform. Moreover, we apply similar optimization techniques used in [BKP20] to compress the proof size from $O(N)$ to $O(\log N)$. Roughly, the prover additionally uses a pseudorandom generator to generate the randomness (i.e., $s', r', \phi, (\mathsf{rand}_i)_{i \in [N]}$). Then, in case $b = 0$, the prover needs to reply only with the seed of size $O(1)$. The prover also uses a Merkle tree to accumulate $(\mathsf{C}_{\phi(i)})_{i \in [N]}$ and sends the root value in the first message. It then only opens to the path necessary for verification when $b = 1$. This has a positive side-effect that we no longer require a permutation ϕ since the path hides the index if we use a slightly tweaked variant of the standard Merkle tree. Finally, we take advantage of the asymmetry in the prover's response size for $b = 0$ and $b = 1$, which are $O(1)$ and $O(\log N)$, respectively. Namely, we imbalance the challenge space so that the prover opens to more 0 than 1, while still maintaining negligible soundness error.

Adding Online-Extractability. To build an accountable ring signature or group signature, we require the above NIZK to be (multi-proof) *online-extractable*. This is a strengthening of standard proof of knowledge (PoK) that roughly states that the knowledge extractor, who can see what the adversary queries to the random oracle, is able to directly extract witnesses from the proofs output by the adversary. The OR-proof by [BKP20], which our NIZK builds on, was only shown to satisfy the standard PoK, which bases on a *rewinding* extractor.

One simple way to add online-extractability to our NIZK is to apply the Unruh transform [Unr15]. Namely, we can modify the prover to add two more commitments $h_0 = \mathsf{Com}(s' \| r', \mathsf{rand}_0)$ and $h_1 = \mathsf{Com}(s'' \| r'', \mathsf{rand}_1)$ in the first message, where Com is instantiated by the random oracle. Then, if $b = 0$ (resp. $b = 1$), the prover further opens to h_0 (resp. h_1). Recall that if the reduction obtains both (s', r') and (s'', r''), then it can invoke the extractor provided by the underlying sigma protocol to extract some (i^*, s^*) such that $(E_0, E_{i^*} = s^* \star E_0) \in \mathsf{R}$. Therefore, for the cheating adversary to fool the reduction, it must guess the bit b and create h_b correctly while creating h_{1-b} arbitrary. Intuitively, if we have λ-repetition of the sigma protocol, then the cheating prover cannot possibly guess all the challenge bits correctly. Therefore, there must be some challenge where it created h_0 and h_1 honestly. For that challenge bit, the reduction algorithm can then retrieve the corresponding inputs $(s' \| r', \mathsf{rand}_0)$ and $(s'' \| r'', \mathsf{rand}_1)$ from simply observing the random

---

[4] The choice of what to encrypt is rather arbitrary. The same idea works if for instance we hash vk into $\mathcal{C}\ell(\mathcal{O})$ and view the digest as the message.

oracle, and then, run the extractor to obtain the witness.

This idea works but it comes with an extra two hashes per one execution of the binary-challenge sigma protocol. Although it may sound insignificant in an asymptotic sense, these hashes add up when we execute the sigma protocol many times, and it makes it difficult to apply some of the optimization tricks. Concretely, when we apply this change to the isogeny-based ring signature by Beullen et al. [BKP20], the signature grows by roughly a factor of 2 to 3.

In this work, we show that we can in fact prove online-extractability *without* making any modification to the aforementioned NIZK. Our main observations are the following: if the prover uses a seed to generate the randomness used in the first message via a random oracle, then the online extractor can observe $(s', r', \phi, (\mathsf{rand}_i)_{i\in[N]})$; and the prover must respond to some execution of the binary-challenge sigma protocol where the challenge bit is 1. The first implies that the seed implicitly acts as a type of commitment to (s', r'). The second implies the prover returns a response that includes (s'', r''). Specifically, our online extractor only looks at all the responses for the rounds where the challenge bit was 1, and checks the random oracle for any seed that leads to the commitment provided in the first message of the sigma protocol. If such seed is found, then it succeeds in extracting a witness. The intuition is simple but it turns out that the formal proof is technically more complicated due to the several optimizations performed on the basic sigma protocol to achieve proof size $O(\log N)$.

Generalizing with Group Actions. Although we have been explaining our generic construction using the CSIDH group action, it is not unique to them. It works equally well for any group action that naturally induces a PKE. Specifically, we instantiate the above idea also by the MLWE group action defined roughly as $\star : R_q^{n+m} \times R_q^m : (\mathbf{s}, \mathbf{e}) \star \mathbf{t} \rightarrow \mathbf{A} \star \mathbf{s} + \mathbf{e} + \mathbf{t}$, where $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. Since CSIDH and MLWE induce a PKE with slightly different algebraic structures, we introduce a *group-action-based* PKE defined by two group actions to formally capture both instances. This abstraction may be of an independent interest since on first glance, isogeny-based and lattice-based PKEs seem to rely on different algebraic structures. Finally, one interesting feature unique to our generic construction is that since our sigma protocol is rather combinatorial in nature, we can for instance use CSIDH for the user's public key vk and mix it with an MLWE-based PKE for the opener' public key opk. The practical impact of such mixture is that we can achieve stronger bit-security for anonymity (due to MLWE) while keeping the user's public key and signature small (due to CSIDH).

Achieving Tight Reduction. Since the proofs do not rely on the forking lemma [FS87, PS00] to extract witnesses from the forged proofs, our construction achieves a tighter reduction compared to prior works on efficient group signatures. However, we still lose a factor $1/N$ in the proof of unforgeability, which may vary from $1/2$ to $1/2^{20}$.[5] Recall N is the size of the group in group signatures but it is the size of all the users enrolled in the system for accountable ring signatures, which may be far larger than the size of the ring. The main reason for this loss was because the reduction needs to guess one user's verification key used by the adversary to create its forgery and to embed the hard problem into it.

A well known technique to obtain a tight proof is to rely on the Katz-Wang technique [KW03] along with the generic OR-composition of sigma protocols, and rely on a multi-instance version of the hard problem (which are believed to be as difficult as the single-instance version for specific hard problems). Namely, we modify the scheme to assign two verification keys $(\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)})$ to each user. The users will only hold one signing key $\mathsf{sk}^{(b)}$ for $b \in \{1, 2\}$ corresponding to the verification key $\mathsf{vk}^{(b)}$. The user can honestly run the aforementioned sigma protocol where the statement includes $\mathsf{vk}^{(b)}$, and a simulated sigma protocol using the ZK-simulator where the statement includes $\mathsf{vk}^{(3-b)}$. We can then use the sequential OR-proof technique as presented in [AOS02, FHJ20] to bridge these two sigma protocols so that it hides the b.[6]

While this generic transform works, it unfortunately doubles the signature size, which may outweigh the motivation for having a tight reduction. In this work, we present a novel and far cheaper technique tailored to our sigma protocol. The signature size overhead is a mere 512B for our concrete lattice-based

---

[5] We note that we also have some independent looseness in the anonymity proof since we rely on the "multi-challenge" $\mathsf{IND\text{-}CPA}$ security from our PKE. This is handled in a standard way, and this is also why we only achieve a truly tight group signature from lattices and not from isogenies.

[6] We note that it seems difficult to use the parallel OR-proof for our sigma protocol since the challenge space is structured.

instantiation. The key observation is that we can view the set of all users' verification key $(\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)})$ as a ring of size $2N$, rather than a ring of size N where each ring element consists of two verification keys. This observation itself is not yet sufficient since recall that we typically must encrypt some information bound to the signer for traceability, e.g., encrypt the position/index of vk in R, and it is no longer clear what to encrypt when we have two verification keys in the ring. Luckily, it turns out that our sigma protocol can be easily modified with no loss in efficiency to overcome this apparent issue. Details are provided in Sec. 5.3.

Concurrent Works. There are two concurrent and independent works published to the Cryptology ePrint Archive [LD21, CHH$^+$21]. Both of these works obtain isogeny-based group signatures from variants of ring signatures. Lai et al. [LD21] bases their group signature on *revocable* ring signatures and Chung et al. [CHH$^+$21] bases it on accountable ring signatures. While Chug et al. follows the security properties formalized in [XY04], our accountable ring signature follows those formalized in [BCC$^+$16]. Since the security property defined in [BCC$^+$16] is stronger, our resulting group signature satisfies more desirable properties. Concretely, even though the opening algorithm of Chung et al. can output an opening proof, the security guarantees do not achieve the ones defined in [BCC$^+$16]. The major difference comes from the fact that their construction does not have *tracing soundness*. In other words, their signature can be opened to two distinct parties with distinct valid opening proofs.

Tab. 2 gives a comparison among three works in terms of the signature size, anonymity and the manager accountability where the integer N represents the size of the group. The construction of [CHH$^+$21] is marked having partial manager accountability due to the aforementioned reason. Ours is the only scheme that achieves an $O(\log N)$ signature size and CCA anonymity. Our scheme additionally provides a much tighter security since both [LD21, CHH$^+$21] rely on the forking lemma in their security proofs.

Notions	Signature Size	Anonymity	Manager Accountable
[LD21]	$\mathcal{O}(N\log(N))$	CPA	No
[CHH$^+$21]	$\mathcal{O}(N^2)$	CPA	Partially
This Work	$\mathcal{O}(\log(N))$	CCA	Yes

Table 2: Comparison with concurrent works in terms of anonymity and the signature size where the interger N represents the size of the group.

Structure of this paper. We begin in Sec. 2 with some preliminary background on sigma protocols, accountable ring signatures, and other mathematical content which this paper relies on. We then introduce our new, generic constructions of accountable ring signature and dynamic group signature schemes in Sec. 3. These generic constructions are built from various components put forward in the proceeding sections: Sec. 4 defines group-action-based hard instance generators and public-key encryption schemes; Sec. 5 introduces our new "traceable" sigma protocol and proves its security; and Sec. 6 then constructs a NIZK proof system from said sigma protocol through the Fiat-Shamir transform. Finally, Sec. 7 details the instantiation of our schemes from isogenies and lattices.

2 Preliminaries

Notation. We begin by introducing some notation that will be used throughout the paper. For $N \in \mathbb{N}$, we denote by $[N]$ the set $\{1, \ldots, N\}$. We use $\|$ to represent concatenation of two strings. We also use $\{X_i\}_{i\in S}$ to denote the set of elements X_i iterating over all values $i \in S$. For any randomized algorithm A taking as input x, we will write $A(x; r)$ to denote the execution of A on x using the randomness r. With an overload in notation, we write $A(x)$ to denote the set of all possible outputs of A on input x, and $y \in A(x)$ to indicate that there exists a randomness r such that $y = A(x; r)$. Finally, we let $\mathsf{negl}(\lambda)$ be a *negligible* function, i.e.

one dominated by $O(\lambda^{-n})$ for all $n > 0$.

A note on random oracles. Throughout the paper, we instantiate several standard cryptographic primitives, such as pseudorandom number generators (i.e., Expand) and commitment schemes, by hash functions modeled as a random oracle $\mathcal{O}$. We always assume the input domain of the random oracle is appropriately separated when instantiating several cryptographic primitives by one random oracle. With abuse of notation, we may occasionally write for example $\mathcal{O}(\mathsf{Expand} \parallel \cdot)$ instead of $\mathsf{Expand}(\cdot)$ to make the usage of the random oracle explicit. Here, we identify Expand with a unique string when inputting it to $\mathcal{O}$. Finally, we denote by $\mathcal{A}^{\mathcal{O}}$ an algorithm $\mathcal{A}$ that has black-box access to $\mathcal{O}$, and we may occasionally omit the superscript $\mathcal{O}$ for simplicity when the meaning is clear from context.

2.1 Sigma Protocols

A sigma protocol Π_Σ for a NP relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a public-coin three-move interactive protocol between a prover and a verifier that satisfies a specific flavor of soundness and zero-knowledge. The language $\mathcal{L}_R$ is defined as $\{\mathsf{X} | (\mathsf{X}, \mathsf{W}) \in R\}$. As standard with many sigma protocols for a language defined over post-quantum algebraic structures, we relax the soundness notion to only hold for a slightly wider relation $\tilde{R}$ (i.e., $R \subseteq \tilde{R}$), e.g., [FO97, DF02, AJL+12, BCK+14, EK18, BKP20]. That is, a cheating prover may not be using a witness in R but is guaranteed to be using some witness in the wider relation $\tilde{R}$. Below, we consider a sigma protocol in the random oracle model, where the prover and verifier have access to a random oracle similarly to [BKP20].[7]

Definition 2.1 (Sigma Protocol)**.** *A sigma protocol* Π_Σ *for the relations* R *and* $\tilde{R}$ *such that* $R \subseteq \tilde{R}$ *(which are implicitly parameterized by the security parameter* λ*) consists of oracle-calling PPT algorithms* $(P = (P_1, P_2), V = (V_1, V_2))$*, where* V_2 *is deterministic and we assume* P_1 *and* P_2 *share states. Let* ChSet *denote the challenge space. Then,* Π_Σ *has the following three-move flow:*

- *The prover, on input* $(\mathsf{X}, \mathsf{W}) \in R$*, runs* $\mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{X}, \mathsf{W})$ *and sends a* commitment com *to the verifier.*
- *The verifier runs* $\mathsf{chall} \leftarrow V_1^{\mathcal{O}}(1^\lambda)$ *to obtain a random* challenge chall *from* ChSet*, and sends it to the prover.*
- *The prover, given* chall*, runs* $\mathsf{resp} \leftarrow P_2^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})$ *and returns a* response resp *to the verifier. Here, we allow* P_2 *to abort with some probability. In such cases we assign* resp *with a special symbol* $\perp$ *denoting abort.*
- *The verifier runs* $V_2^{\mathcal{O}}(\mathsf{X}, \mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *and outputs* $\top$ *(accept) or* $\perp$ *(reject).*

Here, $\mathcal{O}$ *is modeled as a random oracle and we often drop* $\mathcal{O}$ *from the superscript for simplicity when the meaning is clear from context. We assume* X *is always given as input to* P_2 *and* V_2*, and omit it in the following. The protocol* transcript $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *is said to be valid in case* $V_2(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *outputs* $\top$.

We require a sigma protocol Π_Σ in the random oracle model to satisfy the following standard properties: correctness, high min-entropy, special zero-knowledge and (relaxed) special soundness.

We require the sigma protocol to be correct conditioned on the prover not aborting the protocol. Below, if $\delta = 0$, then it corresponds to the case when the prover never aborts.

Definition 2.2 ($(1-\delta)$-Correctness)**.** *A sigma protocol* Π_Σ *is* $(1-\delta)$-correct *for* $\delta \in [0,1]$ *if for all* $\lambda \in \mathbb{N}$ *and* $(\mathsf{X}, \mathsf{W}) \in R$*, the probability of the prover outputting* $\perp$ *is at most* δ*, and we have*

$$\Pr\left[V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{resp}) = \top \;\middle|\; \begin{array}{c} \mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{X}, \mathsf{W}), \\ \mathsf{chall} \leftarrow V_1^{\mathcal{O}}(1^\lambda), \\ \mathsf{resp} \leftarrow P_2^{\mathcal{O}}(\mathsf{W}, \mathsf{chall}) \ s.t.\ \mathsf{resp} \neq \perp. \end{array} \right] = 1,$$

where the probability is taken over the randomness used by (P, V) *and by the random oracle.*

---

[7] This should not be confused with the random oracle used to compile a sigma protocol into an NIZK proof system.

Definition 2.3 (High Min-Entropy). *We say a sigma protocol* Π_Σ *has* $\alpha(\lambda)$ *min-entropy if for any* $\lambda \in \mathbb{N}$, $(\mathsf{X}, \mathsf{W}) \in R$, *and a possibly computationally-unbounded adversary* $\mathcal{A}$, *we have*

$$\Pr\left[\mathsf{com} = \mathsf{com}' \middle| \mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{X}, \mathsf{W}), \mathsf{com}' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{X}, \mathsf{W})\right] \leq 2^{-\alpha},$$

where the probability is taken over the randomness used by P_1 *and by the random oracle. We say* Π_Σ *has* high min-entropy *if* $2^{-\alpha}$ *is negligible in* λ.

Definition 2.4 (Non-Abort Special Zero-Knowledge). *We say* Π_Σ *is (non-abort) special* zero-knowledge *if there exists a PPT simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle* $\mathcal{O}$ *such that for any* $\lambda \in \mathbb{N}$, *statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R$, $\mathsf{chall} \in \mathsf{ChSet}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most a polynomial number of queries to* $\mathcal{O}$, *we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) = 1]\right| = \mathsf{negl}(\lambda),$$

where $\widetilde{P}$ *is a* non-aborting *prover* $P = (P_1, P_2)$ *run on* (X, W) *with a challenge fixed to* chall *and the probability is taken over the randomness used by* (P, V) *and by the random oracle.*

Below, for the special soundness property, the extraction algorithm is only required to recover a "weaker" witness in $\tilde{R}$ rather than in R used in the real protocol. In many applications, the capability of extracting from this wider relation suffices.

Definition 2.5 (Special Soundness). *We say a sigma protocol* Π_Σ *has (relaxed) special soundness if there exists a PT extraction algorithm* $\mathsf{Extract}$ *such that, given a statement* X *and any two valid transcripts* $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *and* $(\mathsf{com}, \mathsf{chall}', \mathsf{resp}')$ *relative to* X *and such that* $\mathsf{chall} \neq \mathsf{chall}'$, *outputs a witness* W *satisfying* $(\mathsf{X}, \mathsf{W}) \in \tilde{R}$.

2.2 Non-Interactive Zero-Knowledge Proofs of Knowledge in the ROM.

We consider non-interactive zero-knowledge proof of knowledge protocols (or simply NIZK (proof system)) in the ROM. Below, we define a variant where the proof is generated with respect to a label. Although syntactically different, such NIZK is analogous to the notion of signature of knowledge [CL06]

Definition 2.6 (NIZK Proof System). *Let* L *denote a label space, where checking membership can be done efficiently. A non-interactive zero-knowledge (*NIZK*) proof system* Π_{NIZK} *for the relations* R *and* $\tilde{R}$ *such that* $R \subseteq \tilde{R}$ *(which are implicitly parameterized by* λ*) consists of oracle-calling PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W}) \rightarrow \pi/\bot$: *On input a label* $\mathsf{lbl} \in \mathsf{L}$, *a statement and witness pair* $(\mathsf{X}, \mathsf{W}) \in R$, *it outputs a proof* π *or a special symbol* $\bot$ *denoting abort.*

$\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) \rightarrow \top/\bot$: *On input a label* $\mathsf{lbl} \in \mathsf{L}$, *a statement* X, *and a proof* π, *it outputs either* $\top$ *(accept) or* $\bot$ *(reject).*

We require a NIZK proof system in the random oracle model to satisfy the following standard properties: correctness, zero-knowledge, (relaxed) statistical soundness, and online extractability. We assume for simplicity that Verify always outputs $\bot$ in case $\mathsf{lbl} \notin \mathsf{L}$.

Definition 2.7 ($(1-\delta)$-Correctness). *A* NIZK *proof system* Π_{NIZK} *is* $(1-\delta)$*-correct for* $\delta \in [0,1]$ *if for all* $\lambda \in \mathbb{N}$, $\mathsf{lbl} \in \mathsf{L}$, $(\mathsf{X}, \mathsf{W}) \in R$, *the probability of* $\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *outputting* $\bot$ *is at most* δ, *and we have*

$$\Pr\left[\ \mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top \ \middle|\ \begin{array}{c} \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W}), \\ \pi \neq \bot. \end{array}\right] = 1,$$

where the probability is taken over the randomness used by $(\mathsf{Prove}, \mathsf{Verify})$ *and by the random oracle.*

Definition 2.8 (Zero-Knowledge)**.** *Let $\mathcal{O}$ be a random oracle, Π_{NIZK} a* NIZK *proof system, and* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *a zero-knowledge simulator for Π_{NIZK}, consisting of two algorithms* Sim_0 *and* Sim_1 *with a shared state. We say the advantage of an adversary $\mathcal{A}$ against* Sim *is*

$$\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\mathcal{O},\mathsf{Prove}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}_0,\mathcal{S}}(1^\lambda) = 1\right]\right| ,$$

where Prove *and $\mathcal{S}$ are prove oracles that on input* $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *return* $\perp$ *if* $\mathsf{lbl} \notin \mathsf{L} \vee (\mathsf{X}, \mathsf{W}) \notin R$ *and otherwise return* $\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *or* $\mathsf{Sim}_1(\mathsf{lbl}, \mathsf{X})$*, respectively. Moreover, the probability is taken also over the randomness of sampling $\mathcal{O}$.*

We say Π_{NIZK} for R and $\tilde{R}$ is zero-knowledge if there exists a PPT simulator Sim *such that for all (possibly computationally-unbounded) adversary $\mathcal{A}$ making at most polynomially many queries to the random oracle and the prover oracle, we have* $\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*.*

Statistical soundness guarantees that any adversary cannot generate a proof for an invalid statement except with a negligible probability.

Definition 2.9 (Statistical Soundness)**.** *Let $\mathcal{O}$ be a random oracle and Π_{NIZK} a* NIZK *proof system. We say the advantage of an adversary $\mathcal{A}$ against soundness is*

$$\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr\left[\begin{array}{c} \nexists \mathsf{W} : (\mathsf{X}, \mathsf{W}) \in \tilde{R} \wedge \\ \mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top \end{array} \middle| \ (\mathsf{lbl}, \mathsf{X}, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)) \right] ,$$

where the probability is taken also over the randomness of sampling $\mathcal{O}$.

We say the NIZK *proof system Π_{NIZK} for R and $\tilde{R}$ has (relaxed) statistical soundness if for all (possibly computationally-unbounded) adversary $\mathcal{A}$ making at most polynomially many queries to the random oracle, we have* $\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*.*

Online extractability requires the existence of an extraction algorithm which, on input a valid proof π and the list or random-oracle queries made by an adversary, always extract a (relaxed) witness except with a negligible probability.

Definition 2.10 (Multi-Proof Online Extractability)**.** *A* NIZK *proof system Π_{NIZK} is (multi-proof) online extractable if there exists a PPT extractor* OnlineExtract *such that for any (possibly computationally-unbounded) adversary $\mathcal{A}$ making at most polynomially-many queries has at most a negligible advantage in the following game played against a challenger (with access to a random oracle $\mathcal{O}$).*

(i) The challenger prepares empty lists $L_{\mathcal{O}}$ and L_P, and sets flag *to* 0*.*

(ii) $\mathcal{A}$ can make random-oracle, prove, and extract queries an arbitrary polynomial number of times:

  - $(\mathtt{hash}, x)$*: The challenger updates $L_{\mathcal{O}} \leftarrow L_{\mathcal{O}} \cup \{x, \mathcal{O}(x)\}$ and returns $\mathcal{O}(x)$. We assume below that $\mathcal{A}$ runs the verification algorithm after receiving a proof from the prove oracle and before submitting a proof to the extract oracle.*[8]
  - $(\mathtt{prove}, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$*: The challenger returns $\perp$ if $\mathsf{lbl} \notin \mathsf{L}$ or $(\mathsf{X}, \mathsf{W}) \notin R$. Otherwise, it returns $\pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ and updates $L_P \leftarrow L_P \cup \{\mathsf{lbl}, \mathsf{X}, \pi\}$.*
  - $(\mathtt{extract}, \mathsf{lbl}, \mathsf{X}, \pi)$*: The challenger checks if $\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top$ and $(\mathsf{lbl}, \mathsf{X}, \pi) \notin L_P$, and returns $\perp$ if not. Otherwise, it runs $\mathsf{W} \leftarrow \mathsf{OnlineExtract}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi, L_{\mathcal{O}})$ and checks if $(\mathsf{X}, \mathsf{W}) \notin \tilde{R}$, and returns $\perp$ if yes and sets* $\mathsf{flag} = 1$*. Otherwise, if all check passes, it returns* W*.*

(iii) At some point $\mathcal{A}$ outputs 1 *to indicate that it is finished with the game. We say $\mathcal{A}$ wins if* $\mathsf{flag} = 1$*. The advantage of $\mathcal{A}$ is defined as* $\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr[\mathcal{A}\text{ wins}]$ *where the probability is also taken over the randomness used by the random oracle.*

---

[8] This is w.l.o.g., and guarantees that the list $L_{\mathcal{O}}$ is updated with the input/output required to verify the proof $\mathcal{A}$ receives or sends.

Note, importantly, that $\mathsf{OnlineExtract}$ is not given access to the queries $\mathsf{Prove}^{\mathcal{O}}$ makes directly to $\mathcal{O}$. Thus, $\mathsf{OnlineExtract}$ is not guaranteed to return a valid witness W when called with any output of the Prove oracle. The requirement that $(\mathsf{lbl}, \mathsf{X}, \pi) \notin L_P$ ensures that this does not allow the adversary to trivially win the game, and in particular by extension ensures that modifying the label lbl should invalidate any proof obtained from the Prove oracle.

Remark 2.11. *If a* NIZK *proof system* Π_{NIZK} *is (multi-proof) online extractable, it is statistically sound—that is, online extractability implies statistical soundness. This is clear, because if an adversary is able to generate an accepting tuple* $(\mathsf{lbl}, \mathsf{X}, \pi)$ *for which* $\nexists \mathsf{W} : (\mathsf{X}, \mathsf{W}) \in \tilde{R}$ *in the soundness game, then clearly* $(\mathtt{extract}, \mathsf{lbl}, \mathsf{X}, \pi)$ *will allow the adversary to win the online extractability game.*

Remark 2.12 (NIZKs with Labels)**.** *If the label space of the* NIZK *is* $\mathsf{L} = \{\perp\}$*, we say the* NIZK *is without labels (or a plain/unlabelled* NIZK*). In this case, we omit the* lbl *argument from the* Prove *and* Verify *functions for clarity.*

2.3 Public-Key Encryption

We recall the standard multi-challenge IND-CPA security of a public-key encryption (PKE) scheme.

Definition 2.13 (Public-Key Encryption)**.** *A public-key encryption* Π_{PKE} *over a message space* $\mathcal{M}$ *consists of four algorithms* $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *On input the security parameter* 1^λ*, it outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: *On input a public parameter* pp*, it outputs a pair of public key and secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{M}) \to \mathsf{ct}$*: On input a public key* pk_i *and a message* $\mathsf{M} \in \mathcal{M}$*, it outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$ *or* $\perp$: *On input a secret key* sk *and a ciphertext* ct*, it outputs either* $\mathsf{M} \in \mathcal{M}$ *or a special symbol* $\perp \notin \mathcal{M}$*.*

We will denote by $\mathcal{R}$ *the set containing the randomness used by the encryption algorithm* Enc*.*

We omit the standard definition of correctness as we provide a more generalized version in Sec. 3.1, Def. 3.1. Below, we define the standard IND-CPA security extended to the multi-challenge setting. Using a textbook hybrid argument, it is clear that the multi-challenge definition is polynomially related to the standard single-challenge definition. The motivation for introducing the multi-challenge variant is because in some cases, we can show that the two definitions are equally difficult without incurring any reduction loss.

Definition 2.14 (Multi-Challenge IND-CPA Security)**.** *A* PKE *scheme* $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* multi-challenge IND-CPA secure *against* Q *challenges if, for any* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and samples a bit* $b \in \{0, 1\}$*. The challenger provides* $(\mathsf{pp}, \mathsf{pk})$ *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *can adaptively query the challenge oracle at most* Q *times. In each query,* $\mathcal{A}$ *sends a pair of messages* $(\mathsf{M}_0, \mathsf{M}_1) \in \mathcal{M}^2$*, and the challenger returns* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{M}_b)$ *to* $\mathcal{A}$*.*

(iv) $\mathcal{A}$ *outputs a bit* $b^* \in \{0, 1\}$*. We say* $\mathcal{A}$ *wins if* $b^* = b$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}},Q}(\mathcal{A}) = |\Pr[\mathcal{A}\ wins] - 1/2|$.

2.4 Accountable Ring Signatures

We provide the definition of accountable ring signatures (ARSs), following the formalization introduced by Bootle et al. [BCC+15].

Definition 2.15 (Accountable Ring Signature)**.** *An accountable ring signature* Π_{ARS} *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{OKGen}, \mathsf{UKGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open}, \mathsf{Judge})$ *defined as follows:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *On input a security parameter* 1^λ*, it returns a public parameter* pp *(sometimes implicitly) used by the scheme. We assume* pp *defines openers' public-key space* $\mathcal{K}_{\mathsf{opk}}$ *and users' verification-key space* $\mathcal{K}_{\mathsf{vk}}$*, with efficient algorithms to decide membership.*

$\mathsf{OKGen}(\mathsf{pp}) \to (\mathsf{opk}, \mathsf{osk})$: *On input a public parameter* pp*, it outputs a pair of public and secret keys* $(\mathsf{opk}, \mathsf{osk})$ *for an* opener*.*

$\mathsf{UKGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: *On input a public parameter* pp*, it outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$ *for a* user*.*

$\mathsf{Sign}(\mathsf{opk}, \mathsf{sk}, \mathsf{R}, \mathsf{M}) \to \sigma$: *On input an opener's public key* opk*, a signing key* sk*, a list of verification keys, i.e., a* ring*,* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, and a message* M*, it outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) \to \top/\bot$: *On input an opener's public key* opk*, a ring* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, a message* M*, and a signature* σ*, it (deterministically) outputs either* $\top$ *(accept) or* $\bot$ *(reject).*

$\mathsf{Open}(\mathsf{osk}, \mathsf{R}, \mathsf{M}, \sigma) \to (\mathsf{vk}, \pi)/\bot$: *On input an opener's secret key* osk*, a ring* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, a message* M*, a signature* σ*, it (deterministically) outputs either a pair of verification key* vk *and a proof* π *that the owner of* vk *produced the signature, or* $\bot$*.*

$\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) \to \top/\bot$: *On input an opener's public key* opk*, a ring* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, a verification key* vk*, a message* M*, a signature* σ*, and a proof* π*, it (deterministically) outputs either* $\top$ *(accept) or* $\bot$ *(reject). We assume without loss of generality that* $\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi)$ *outputs* $\bot$ *if* $\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma)$ *outputs* $\bot$*.*

An accountable ring signature is required to satisfy the following properties: correctness, anonymity, traceability, unforgeability, and tracing soundness.

First, we require correctness to hold even if the ring contains maliciously-generated user keys or the signature has been produced for a maliciously-generated opener key. Note that the correctness guarantee for the open and judge algorithms are defined implicitly in the subsequent security definitions.

Definition 2.16 (Correctness)**.** *An accountable ring signature* Π_{ARS} *is* correct *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in* λ *in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and generates a user key* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{UKGen}(\mathsf{pp})$*. It then provides* $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *outputs an opener's public key, a ring, and a message tuple* $(\mathsf{opk}, \mathsf{R}, \mathsf{M})$ *to the challenger.*

(iii) The challenger runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{opk}, \mathsf{sk}, \mathsf{R}, \mathsf{M})$*. We say* $\mathcal{A}$ *wins if*

  - $\mathsf{opk} \in \mathcal{K}_{\mathsf{opk}}$, $\mathsf{R} \subseteq \mathcal{K}_{\mathsf{vk}}$, *and* $\mathsf{vk} \in \mathsf{R}$,
  - $\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \bot$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Correct}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A}$ *wins*$]$.

Anonymity requires that a signature does not leak any information on who signed it. We consider the standard type of anonymity notion where the adversary gets to choose the signing key used to generate the signature. Moreover, we allow the adversary to make (non-trivial) opening queries that reveal who signed the messages. This notion is often called *full* (*CCA*) anonymity [BMW03, BCC+16] to differentiate between weaker notions of anonymity such as *selfless* anonymity that restricts the adversary from exposing the signing key used to sign the signature or *CPA* anonymity where the adversary is restricted from querying the open oracle.

Definition 2.17 (Anonymity)**.** *An accountable ring signature* Π_{ARS} *is (CCA)* anonymous (against full key exposure) *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the following game played against a challenger.*

(i) *The challenger runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and generates an opener key* $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OKGen}(\mathsf{pp})$*. It also prepares an empty list* $\mathsf{Q}_{\mathtt{sign}}$ *and samples a random bit* $b \leftarrow \{0, 1\}$*.*

(ii) *The challenger provides* $(\mathsf{pp}, \mathsf{opk})$ *to* $\mathcal{A}$*.*

(iii) $\mathcal{A}$ *can make signing and opening queries an arbitrary polynomial number of times:*

- $(\mathtt{sign}, \mathsf{R}, \mathsf{M}, \mathsf{sk}_0, \mathsf{sk}_1)$*: The challenger runs* $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{opk}, \mathsf{sk}_i, \mathsf{R}, \mathsf{M})$ *for* $i \in \{0, 1\}$ *and returns* $\perp$ *if* $\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma_i) = \perp$ *for either of* $i \in \{0, 1\}$*. Otherwise, it updates* $\mathsf{Q}_{\mathtt{sign}} \leftarrow \mathsf{Q}_{\mathtt{sign}} \cup \{(\mathsf{R}, \mathsf{M}, \sigma_b)\}$ *and returns* σ_b*.*
- $(\mathtt{open}, \mathsf{R}, \mathsf{M}, \sigma)$*: The challenger returns* $\perp$ *if* $(\mathsf{R}, \mathsf{M}, \sigma) \in \mathsf{Q}_{\mathtt{sign}}$*. Otherwise, it returns* $\mathsf{Open}(\mathsf{osk}, \mathsf{R}, \mathsf{M}, \sigma)$*.*

(iv) $\mathcal{A}$ *outputs a guess* b^**. We say* $\mathcal{A}$ *wins if* $b^* = b$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$*.*

Unforgeability considers two types of forgeries. The first captures the natural notion of unforgeability where an adversary cannot forge a signature for a ring of honest users, i.e., a ring of users for which it does not know any of the corresponding secret keys. The second captures the fact that an adversary cannot accuse an honest user of producing a signature even if the ring contains malicious users and the opener is malicious.

Definition 2.18 (Unforgeability)**.** *An accountable ring signature scheme* Π_{ARS} *is* unforgeable (with respect to insider corruption) *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) *The challenger runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and initializes an empty keyed dictionary* $\mathsf{D}_{\mathtt{UKey}}[\cdot]$ *and three empty sets* $\mathsf{Q}_{\mathtt{UKey}}$*,* $\mathsf{Q}_{\mathtt{sign}}$ *and* $\mathsf{Q}_{\mathtt{cor}}$*. It provides* pp *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *can make user key generation, signing, and corruption queries an arbitrary polynomial number of times:*

- $(\mathtt{ukeygen})$*: The challenger runs* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{UKGen}(\mathsf{pp})$*. If* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}] \neq \perp$*, then it returns* $\perp$*. Otherwise, it updates* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}] = \mathsf{sk}$ *and* $\mathsf{Q}_{\mathtt{UKey}} \leftarrow \mathsf{Q}_{\mathtt{UKey}} \cup \{\mathsf{vk}\}$*, and returns* vk*.*
- $(\mathtt{sign}, \mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M})$*: The challenger returns* $\perp$ *if* $\mathsf{vk} \notin \mathsf{Q}_{\mathtt{UKey}} \cap \mathsf{R}$*. Otherwise, it runs* $\sigma \leftarrow \mathsf{Sign}(\mathsf{opk}, \mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}], \mathsf{R}, \mathsf{M})$*. The challenger updates* $\mathsf{Q}_{\mathtt{sign}} \leftarrow \mathsf{Q}_{\mathtt{sign}} \cup \{(\mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma)\}$ *and returns* σ*.*
- $(\mathtt{corrupt}, \mathsf{vk})$*: The challenger returns* $\perp$ *if* $\mathsf{vk} \notin \mathsf{Q}_{\mathtt{UKey}}$*. Otherwise, it updates* $\mathsf{Q}_{\mathtt{cor}} \leftarrow \mathsf{Q}_{\mathtt{cor}} \cup \{\mathsf{vk}\}$ *and returns* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}]$*.*

(iv) $\mathcal{A}$ *outputs* $(\mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma, \pi)$*. We say* $\mathcal{A}$ *wins if*

  – $(\mathsf{opk}, *, \mathsf{R}, \mathsf{M}, \sigma) \notin \mathsf{Q}_{\mathtt{sign}}$*,* $\mathsf{R} \subseteq \mathsf{Q}_{\mathtt{UKey}} \backslash \mathsf{Q}_{\mathtt{cor}}$*,*

  - $\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \top$,

or

  - $(\mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma) \notin \mathsf{Q}_{\mathtt{sign}}$, $\mathsf{vk} \in \mathsf{Q}_{\mathtt{UKey}} \backslash \mathsf{Q}_{\mathtt{cor}}$,
  - $\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) = \top$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

Traceability requires that any opener key pair $(\mathsf{opk}, \mathsf{osk})$ in the range of the opener key-generation algorithm can open a valid signature σ to some user vk along with a proof valid π. This ensures that any opener can trace the user and produce a proof for its decision. Below, rather than assuming an efficient algorithm that checks set membership $(\mathsf{opk}, \mathsf{osk}) \in \mathsf{OKGen}(\mathsf{pp})$, we simply ask the adversary to output the randomness used to generate $(\mathsf{opk}, \mathsf{osk})$. Note that this definition contains the prior definitions where opk was assumed to be uniquely defined and efficiently computable from osk [BCC+15].

Definition 2.19 (Traceability). *An accountable ring signature scheme* Π_{ARS} *is* traceable *if, for all* $\lambda \in \mathbb{N}$, *any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and provides* pp *to* $\mathcal{A}$.

(ii) $\mathcal{A}$ *returns a randomness, a ring, a message, and a signature tuple* $(\mathsf{rr}, \mathsf{R}, \mathsf{M}, \sigma)$. *We say* $\mathcal{A}$ *wins if*

  - $\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \top$, *where* $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OKGen}(\mathsf{pp}; \mathsf{rr})$, *and*
  - $\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) = \bot$, *where* $(\mathsf{vk}, \pi) \leftarrow \mathsf{Open}(\mathsf{osk}, \mathsf{R}, \mathsf{M}, \sigma)$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

Finally, tracing soundness requires that a signature cannot trace to two different users in the ring. This must hold even if all the users in the ring and the opener are corrupt.

Definition 2.20 (Tracing Soundness). *An accountable ring signature scheme* Π_{ARS} *is* traceable sound *if, for all* $\lambda \in \mathbb{N}$, *any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and provides* pp *to* $\mathcal{A}$.

(ii) $\mathcal{A}$ *returns an opener's public key, a ring, a message, a signature, and two verification keys and proofs* $(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma, \{(\mathsf{vk}_b, \pi_b)\}_{b \in \{0,1\}})$. *We say* $\mathcal{A}$ *wins if*

  - $\mathsf{vk}_0 \neq \mathsf{vk}_1$,
  - $\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}_0, \mathsf{M}, \sigma, \pi_0) = \top$,
  - $\mathsf{Judge}(\mathsf{opk}, \mathsf{R}, \mathsf{vk}_1, \mathsf{M}, \sigma, \pi_1) = \top$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

2.5 Isogenies and Ideal Class Group Actions

Let $\mathbb{F}_p$ be a prime field, with $p \geq 5$. In the following E and E' denote elliptic curves defined over $\mathbb{F}_p$. An isogeny $\varphi : E \to E'$ is a non-constant morphism mapping 0_E to $0_{E'}$. Each coordinate of $\varphi(x, y)$ is then the fraction of two polynomials in $\overline{\mathbb{F}}_p[x, y]$, where $\overline{\mathbb{F}}_p$ denotes the algebraic closure of $\mathbb{F}_p$. If the coefficients of the polynomials lie in $\mathbb{F}_p$, then φ is said to be defined over $\mathbb{F}_p$. We restrict our attention to *separable* isogenies (which induce separable extensions of function fields) between *supersingular* elliptic curves defined over $\mathbb{F}_p$, i.e., curves whose set of rational points $E(\mathbb{F}_p)$ has cardinality $p + 1$.

An isogeny $\varphi : E \to E'$ is an *isomorphism* if its kernel is equal to $\{0_E\}$, and an *endomorphism* of E if $E = E'$. The set $\mathrm{End}_p(E)$ of all endomorphisms of E that are defined over $\mathbb{F}_p$, together with the zero map, form a commutative ring under pointwise addition and composition. $\mathrm{End}_p(E)$ is isomorphic to an order $\mathcal{O}$ of the quadratic field $\mathbb{K} = \mathbb{Q}(\sqrt{-p})$ [CLM+18]. We recall that an order is a subring of $\mathbb{K}$, which is also a finitely-generated $\mathbb{Z}$-module containing a basis of $\mathbb{K}$ as a $\mathbb{Q}$-vector space. A fractional ideal $\mathfrak{a}$ of $\mathcal{O}$ is a finitely generated $\mathcal{O}$-submodule of $\mathbb{K}$. We say that $\mathfrak{a}$ is invertible if there exists another fractional ideal $\mathfrak{b}$ of $\mathcal{O}$ such that $\mathfrak{a}\mathfrak{b} = \mathcal{O}$, and that it is principal if $\mathfrak{a} = \alpha\mathcal{O}$ for some $\alpha \in \mathbb{K}$. The invertible fractional ideals of $\mathcal{O}$ form an Abelian group whose quotient by the subgroup of principal fractional ideals is finite. This quotient group is called the *ideal class group* of $\mathcal{O}$, and denoted by $\mathcal{C}\ell(\mathcal{O})$.

The ideal class group $\mathcal{C}\ell(\mathcal{O})$ acts freely and transitively on the set $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, which contains all supersingular elliptic curves E over $\mathbb{F}_p$ - modulo isomorphisms defined over $\mathbb{F}_p$ - such that there exists an isomorphism between $\mathcal{O}$ and $\mathrm{End}_p(E)$ mapping $\sqrt{-p} \in \mathcal{O}$ into the Frobenius endomorphism $(x, y) \mapsto (x^p, y^p)$. We denote this action by $*$. Recently, it has been used to design several cryptographic primitives [CLM+18, DG19, BKV19, LGd21], whose security proofs rely on (variations of) the Group Action Inverse Problem (GAIP), defined as follows.

Definition 2.21 (Group Action Inverse Problem (GAIP))**.** *Let $[E_0]$ be an element in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where p is an odd prime and $\mathcal{O}$ an order in $\mathbb{Q}(\sqrt{-p})$. Given $[E]$ sampled from the uniform distribution over $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, the GAIP_p problem consists in finding an element $[\mathfrak{a}] \in \mathcal{C}\ell(\mathcal{O})$ such that $[\mathfrak{a}] * [E_0] = [E]$.*

The best known classical algorithm to solve the GAIP problem has time complexity $O(\sqrt{N})$, where $N = |\mathcal{C}\ell(\mathcal{O})|$. The best known quantum algorithm, on the other hand, is Kuperberg's algorithm for the hidden shift problem [Kup05, Kup13]. It has a subexponential complexity, for which the concrete security estimates are still an active area of research [BLMP19, Pei20, BS20, CSCDJRH20].

For the security of the isogeny-based instantiations, we will also rely on a multi-instance variant the GAIP problem which is trivially equivalent to the GAIP problem.

Definition 2.22 (Multi-Instance GAIP (MI-GAIP) Problem)**.** *Let $[E_0]$ be an element in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where p is an odd prime and $\mathcal{O}$ an order in $\mathbb{Q}(\sqrt{-p})$. Given $[E_1], \cdots, [E_N]$ sampled uniformly at random from $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where $N \in \mathbb{N}$, the $\mathsf{MI\text{-}GAIP}_{p,N}$ problem consists in finding an element $[\mathfrak{a}] \in \mathcal{C}\ell(\mathcal{O})$ such that $[\mathfrak{a}] * [E_0] = [E_i]$ for some $i \in [N]$.*

To see the equivalence (informally), given an instance of the GAIP problem $([E_0], [E])$, sample $[\mathfrak{r}_1], \ldots, [\mathfrak{r}_N] \in \mathcal{C}\ell(\mathcal{O})$, and compute $[E_i] = [\mathfrak{r}_i] * [E]$ for each i. Then a solution for the MT-GAIP on $([E_0], [E_1], \ldots, [E_N])$, say $[\mathfrak{a}] * [E_0] = [E_j]$, results in a solution to the GAIP by computing $[\mathfrak{a}][\mathfrak{r}_j]^{-1}$.

We also need the following assumption, the decisional CSIDH Problem. Looking ahead, the distinguishing problems will ensure (multi-instance) IND-CPA for our PKE in Sec. 7.1 and therefore anonymity for our ring/group signature schemes. Note that we will require the class group to be of odd order to avoid the attack presented in [CSV20]. Equivalently, we require $p = 3 \mod 4$.

Definition 2.23 (Decisional CSIDH (dCSIDH) Problem)**.** *Let $[E_0]$ be an element in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where p is an odd prime. The decisional CSIDH problem is that given a tuple $([\mathfrak{a}_1] * [E_0], [\mathfrak{a}_2] * [E_0], E)$ where $[\mathfrak{a}_1], [\mathfrak{a}_2]$ are sampled uniformly from $\mathcal{C}\ell(\mathcal{O})$ and $[E]$ is either sampled uniformly from $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ or $[E] = [\mathfrak{a}_1\mathfrak{a}_2] * [E_0]$, and decide which distribution $[E]$ is drawn from.*

2.6 Lattices

Let R and R_q denote the rings $\mathbb{Z}[X]/(X^n + 1)$ and $\mathbb{Z}[X]/(q, X^n + 1)$ for integers n and q, respectively. Norms over R are defined through the coefficient vectors of the polynomials, which lie over $\mathbb{Z}^n$. Norms over R_q are defined in the conventional way by uniquely representing coefficients of elements over R_q by elements in the range $(-q/2, q/2]$ when q is even and $[-(q-1)/2, (q-1)/2]$ when q is odd (see for example [DKL+18] for more details).

The hard problems we will rely on are the *module short integer solution* (MSIS) problem and *module learning with errors* (MLWE) problem, first introduced in [LS15].

Definition 2.24 (Module Short Integer Solution)**.** *Let n, q, k, ℓ, γ be integers. The advantage for the (Hermite normal form)* module short integer solution problem $\mathsf{MSIS}_{n,q,k,\ell,\gamma}$ *for an algorithm $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,\gamma}(\mathcal{A}) = \Pr\left[\begin{array}{c} 0 < \|\mathbf{u}\|_\infty \le \gamma \wedge \\ [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{u} = \mathbf{0} \end{array} \middle| \mathbf{A} \leftarrow R_q^{k\times\ell}, \mathbf{u} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})\right].$$

Definition 2.25 (Module Learning with Errors)**.** *Let n, q, k, ℓ be integers and D a probability distribution over R_q. For any $\mathbf{A} \in R_q^{k\times\ell}$, define two oracles as follows:*

- *$\mathcal{O}_{\mathbf{A}}$: Sample $(\mathbf{s}, \mathbf{e}) \leftarrow D^k \times D^\ell$ and output $\mathbf{A}\mathbf{s} + \mathbf{e} \in R_q^k$,*
- *$\mathcal{O}_\$$: Output a random $\mathbf{b} \leftarrow R_q^k$.*

The advantage for the decision module learning with errors problem $\mathsf{sMLWE}_{n,q,k,\ell,D}$ *for an algorithm $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathsf{dMLWE}}_{n,q,k,\ell,D}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{A}}}(1^\lambda, \mathbf{A}) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}_\$}(1^\lambda, \mathbf{A}) \to 1]\right|,$$

where the probability is taken also over the random choice of $\mathbf{A} \leftarrow R_q^{k\times\ell}$.

The advantage for the search learning with errors problem $\mathsf{sMLWE}_{n,q,k,\ell,D}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{sMLWE}}_{n,q,k,\ell,D}(\mathcal{A}) = \Pr\left[\begin{array}{c} \mathbf{v} = \mathbf{A}\mathbf{s} + \mathbf{e} \wedge \\ (\mathbf{s}, \mathbf{e}) \in \mathsf{Supp}(D^\ell) \times \mathsf{Supp}(D^k) \end{array} \middle| (\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathbf{A}}}(1^\lambda, \mathbf{A})\right],$$

where $\mathbf{v}$ is one of the vectors returned by $\mathcal{O}_{\mathbf{A}}$.

In this work, we consider the MLWE problem where an adversary is given oracle access to a MLWE sample generator. For any PPT adversary $\mathcal{A}$, this is polynomially related to the conventional single-instance MLWE problem via a standard hybrid argument. There is also a simple *tight* reduction from the single-instance to the multi-instance MLWE problem *à la* "noise-flooding," where (roughly) the support of the distribution D considered by the multi-instance problem is required to be super-polynomially larger than those considered by the single-instance problem. However, practically speaking, to the best of our knowledge, we are not aware of any attacks that exploit the multiplicity of the MLWE sample. Therefore, throughout this work, we assume the multi-instance MLWE problem to be as difficult as the single-instance MLWE problem.

The assumption on the hardness of (multi-instance) MLWE is believed to hold even when D is the uniform distribution over ring elements with infinity norm at most a fixed value B, say $B \approx 5$, for appropriate choices of n, q, k, ℓ [ACD+18]. We write $\mathsf{MLWE}_{n,q,k,\ell,B}$ when we consider such distribution D. For example, the round-2 NIST candidate signature scheme Dilithium [DKL+18] uses such parameters for the (single-instance) MLWE problem, and in particular, our scheme borrows the same parameter sets.

3 Generic Construction of Accountable Ring Signature and Dynamic Group Signature

In this section, we present novel generic frameworks for accountable ring signature, dynamic group signature, and their tightly secure variants. Firstly, we introduce a generic construction of an accountable ring signature in Sec. 3.1. Constructing a dynamic group signature immediately follows by limiting the functionality of accountable ring signature. Our construction achieves a tighter reduction compared to prior works on efficient group signatures as it does not rely on the forking lemma [FS87, PS00]. However, since we still lose a factor of $1/N$ in the reduction, we finally show how to modify our construction to be truly tight using the Katz-Wang technique [KW03] in Sec. 3.3.

3.1 Generic Construction of Accountable Ring Signature

In this subsection, we present our generic construction of an accountable ring signature scheme. Before diving in the details we give a brief overview of our generic construction. The setup is as follows. The opening authorities generate a PKE key-pair, denoted as $(\mathsf{opk}, \mathsf{osk})$ to indicate that they are the opener's keys, and publish the opening public key opk. The users generate an element (x, w) in a hard relation R, and publish the statement x as verification key, and keep the witness w as secret signing key. A signature for our ARS scheme for a ring $R = \{\mathsf{x}_1, \ldots, \mathsf{x}_N\}$ consists of a ciphertext ct, and a NIZK proof that: 1) The ciphertext is an encryption of an index $I \in [N]$ under an opener public key opk, and 2) that the signer knows a witness w corresponding to the I-th statement x_I in the ring R. The second property ensures that the signature is unforgeable, and the first property ensures that the opener (who has the secret key opk) can decrypt the ciphertext to find out who the real signer is. To convince others that a signature was produced by the I-th member of the ring, the opener uses a second NIZK proof to prove that he knows a opener secret key osk that is consistent with opk, and such that $\mathsf{Dec}(\mathsf{osk}, \mathsf{ct}) = I$. If the opener could find a second secret key osk', consistent with opk and such that ct decrypts to $I' \neq I$ under osk', then the opener could frame I' for signing a signature, which breaks the tracing soundness of the signature scheme. To prevent this we require the PKE to satisfy a strong correctness property, which says that an encryption of I will always decrypt to I, even if the encryption randomness and decryption key are invalid (in some specific, controlled way). More formally we define the following special correctness notion for a PKE scheme.

Definition 3.1 (($\mathcal{R}', \mathcal{KR}'$)-correctness)**.** *Consider a public-key encryption scheme* $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, with* $\mathcal{R}$ *the set containing all possible randomness used by* Enc *and* $\mathcal{KR}$ *the binary relation that contains all the key pairs* $(\mathsf{pk}, \mathsf{sk})$ *that can be generated by running* KeyGen*. Let* $\mathcal{R}'$ *be a set containing* $\mathcal{R}$*, and* $\mathcal{KR}'$ *a relation containing* $\mathcal{KR}$*. Then we say that* Π_{PKE} *is* $(\mathcal{R}', \mathcal{KR}')$*-correct if, for all* $\lambda \in \mathbb{N}$*, and for all but a negligible fraction of* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, we have for all* $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR}'$*, for all messages* m *in the plaintext space* $\mathcal{M}$*, and all* $r \in \mathcal{R}'$ *that*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) = m\,.$$

Remark 3.2. *Note that* pp *is also implicitly used in the relations* $\mathcal{KR}, \mathcal{KR}'$*. If* $\mathcal{R}' = \mathcal{R}$ *and* $\mathcal{KR}' = \mathcal{KR}$*, then the* $(\mathcal{R}', \mathcal{KR}')$*-correctness is exactly the standard correctness property for* PKE*s. If* $\mathcal{R}'$ *or* $\mathcal{KR}'$ *is larger than* $\mathcal{R}$ *or* $\mathcal{KR}$*, respectively, then the definition becomes a stronger property, because the decryption algorithm is required to decrypt correctly even when the encryption algorithm used some invalid randomness, and/or when the keypair is invalid. (*$\mathcal{R}'$ *and* $\mathcal{KR}'$ *control how "invalid" randomness and secret key are allowed to be.)*

Our generic construction of an accountable ring signature scheme $\Pi_{\mathsf{ARS}} = (\mathsf{ARS.Setup}, \mathsf{ARS.OKGen}, \mathsf{ARS.UKGen}, \mathsf{ARS.Sign}, \mathsf{ARS.Verify}, \mathsf{ARS.Open}, \mathsf{ARS.Judge})$, provide in Fig. 1, is based on the following building blocks:

- A hard-instance generator contains a setup algorithm $\mathsf{RelSetup}$ that, on input a security parameter λ, outputs a description pp of a pair of binary relations $R_{\mathsf{pp}} \subseteq \tilde{R}_{\mathsf{pp}}$, and a instance generator IGen for those pairs of relations. That is, $\mathsf{RelSetup}$ and IGen are PPT algorithms such that $\Pr[(\mathsf{x}, \mathsf{w}) \in R_{\mathsf{pp}} \mid \mathsf{pp} \leftarrow \mathsf{RelSetup}(1^\lambda); (\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{IGen}(\mathsf{pp})] = 1$, and such that if we define the advantage of an adversary $\mathcal{A}$ against $(\mathsf{RelSetup}, \mathsf{IGen})$ as
$$\mathsf{Adv}^{\mathsf{Hard}}_{\mathsf{RelSetup},\mathsf{IGen}}(\mathcal{A}) = \Pr\left[(\mathsf{x}, \mathsf{w}') \in \tilde{R}_{\mathsf{pp}} \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{RelSetup}(1^\lambda) \\ (\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{IGen}(\mathsf{pp}) \\ \mathsf{w}' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{x}) \end{array} \right],$$
then $\mathsf{Adv}^{\mathsf{Hard}}_{\mathsf{RelSetup},\mathsf{IGen}}(\mathcal{A})$ is a negligible function of λ for every PPT adversary $\mathcal{A}$.
- A public-key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with multi-challenge IND-CPA security, and with $(\mathcal{R}', \mathcal{KR}')$-correctness for some relaxed randomness set $\mathcal{R}'$ and some relaxed key relation $\mathcal{KR}'$. The message space of the encryption scheme contains a set of indices $[N]$ for any polynomially large $N \in \mathbb{N}$.

- A multi-proof online extractable NIZK proof system with labels $\Pi_{\mathsf{NIZK,lbl}} = (\mathsf{NIZK.Setup}_{\mathsf{lbl}}, \mathsf{NIZK.Prove}_{\mathsf{lbl}}, \mathsf{NIZK.Verify}_{\mathsf{lbl}})$ for the relations

  $$R_{\mathsf{sig}} = \left\{ \left((\{\mathsf{x}_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, \mathsf{w}, r) \right) \;\middle|\; (\mathsf{x}_I, \mathsf{w}) \in R_{\mathsf{pp}} \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \right\}$$
  $$\tilde{R}_{\mathsf{sig}} = \left\{ \left((\{\mathsf{x}_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, \mathsf{w}, r) \right) \;\middle|\; (\mathsf{x}_I, \mathsf{w}) \in \tilde{R}_{\mathsf{pp}} \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \right\} .$$

  To be precise, we need to also include the public parameters output by $\mathsf{RelSetup}$ and $\mathsf{PKE.Setup}$ in the statement. We omit them for better readability.

- A statistically sound NIZK proof system (without labels) $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the relations

  $$R_{\mathsf{open}} = \{((\mathsf{pk}, \mathsf{ct}, I), \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR} \wedge \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = I\}$$
  $$\tilde{R}_{\mathsf{open}} = \left\{((\mathsf{pk}, \mathsf{ct}, I), \mathsf{sk}) \;\middle|\; (\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR}' \wedge \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = I\right\} .$$

  Similarly to above, we omit the public parameter output by $\mathsf{PKE.Setup}$ in the statement. We emphasize that Π_{NIZK} does not need to be online extractable.

```
ARS.Setup($1^\lambda$)
  $\mathsf{pp}_1 \leftarrow \mathsf{RelSetup}(1^\lambda)$
  $\mathsf{pp}_2 \leftarrow \mathsf{PKE.Setup}(1^\lambda)$
  return $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$
```

```
ARS.UKGen(pp)
  $(\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{IGen}(\mathsf{pp}_1)$
  return $(\mathsf{vk} := \mathsf{x}, \mathsf{sk} := \mathsf{w})$
```

```
ARS.Verify(opk, R, M, $\sigma$)
  $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  return $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\mathtt{sign}})$
```

```
ARS.Judge(opk, R, vk, M, $\sigma$, $\pi_{\mathsf{open}}$)
  $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  if $\nexists I : \mathsf{vk} = \mathsf{R}_I$ then
      return $\bot$.
  $b_0 \leftarrow \mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma)$
  $b_1 \leftarrow \mathsf{NIZK.Verify}((\mathsf{opk}, \mathsf{ct}, I), \pi_{\mathsf{open}})$
  return $b_0 \wedge b_1$
```

```
ARS.OKGen(pp)
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_2)$
  return $(\mathsf{opk} := \mathsf{pk}, \mathsf{osk} := \mathsf{sk})$
```

```
ARS.Sign(opk, sk, R, M)
  $\{\mathsf{x}_i\}_{i\in[N]} \leftarrow \mathsf{R}$
  if $\nexists I : (\mathsf{x}_I, \mathsf{sk}) \in R_{\mathsf{pp}_1}$ then
      return $\bot$.
  $r \xleftarrow{\$} \mathcal{R}$
  $\mathsf{ct} = \mathsf{Enc}(\mathsf{opk}, I; r)$
  $\pi_{\mathtt{sign}} \leftarrow \mathsf{NIZK.Prove}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r))$
  return $\sigma := (\mathsf{ct}, \pi_{\mathtt{sign}})$
```

```
ARS.Open(osk, R, M, $\sigma$)
  if $\mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \bot$ then
      return $\bot$
  $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  $I \leftarrow \mathsf{Dec}(\mathsf{osk}, \mathsf{ct})$
  $\pi_{\mathsf{open}} \leftarrow \mathsf{NIZK.Prove}((\mathsf{opk}, \mathsf{ct}, I), \mathsf{osk})$
  return $\pi := (\mathsf{R}_I, \pi_{\mathsf{open}})$
```

Figure 1: Generic construction of an accountable ring signature Π_{ARS} obtained from a hard instance generator $(\mathsf{RelSetup}, \mathsf{IGen})$, a public-key encryption algorithm $(\mathsf{PKE.Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfying some suitable security and correctness properties, a NIZK with labels $\Pi_{\mathsf{NIZK,lbl}}$ for R_{sig}, and a NIZK without labels Π_{NIZK} for R_{open}. The public parameter pp is provided to all algorithms where we may omit them for readability.

Correctness and security of the proposed accountable ring signature scheme Π_{ARS} are shown in the following theorems.

Theorem 3.3. *The accountable ring signature scheme Π_{ARS} in Fig. 1 is correct.*

Proof. Due to the correctness of the underlying NIZK proof system, $\Pi_{\mathsf{NIZK,lbl}}$, any signature output by $\mathsf{ARS.Sign}$ will be accepted by $\mathsf{ARS.Verify}$ with probability 1. ☐

Theorem 3.4. *The accountable ring signature scheme Π_{ARS} in Fig. 1 is (CCA) anonymous (against full key exposure) in the random oracle model, assuming Π_{PKE} is multi-challenge* $\mathsf{IND\text{-}CPA}$ *secure and* $(\mathcal{R}', \mathcal{KR}')$*-correct,* $\Pi_{\mathsf{NIZK,lbl}}$ *is zero-knowledge, multi-challenge online-extractable, and* Π_{NIZK} *is zero-knowledge. Precisely, for an adversary $\mathcal{A}$, running in time T, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, with running times $O(T)$ such that*

$$\mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_3) + \mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{B}_4)\,.$$

Proof. We prove anonymity using a hybrid argument with the following series of games. Let the advantage of the adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

Game_1 : This is the original anonymity game defined in Def. 2.17. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$ by definition.

Game_2 : This is the same as Game_1, except that it uses the simulator $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$ for Π_{NIZK} to answer random-oracle and opening queries from the adversary. When $\mathcal{A}$ makes a random oracle query, the challenger forwards the query to $\mathsf{NIZK.Sim}_0$, records the query and answers, and forwards the answer to $\mathcal{A}$. When $\mathcal{A}$ makes an opening query, rather than computing π_{open} using $\mathsf{NIZK.Prove}$ and osk, the challenger instead uses the output of $\mathsf{NIZK.Sim}_1$. We consider an adversary $\mathcal{B}_1$ against the zero-knowledge property of Π_{NIZK} which simulates Game_2 for $\mathcal{A}$. Let Prove and $\mathcal{S}$ be as in the definition of zero-knowledge for the NIZK proof system. Then, if $\mathcal{B}_1's$ oracle queries are answered by $(\mathcal{O}, \mathsf{Prove})$ the game is identical to Game_1, and if queries are answered by $(\mathsf{NIZK.Sim}_0, \mathcal{S})$, then the game is identical to Game_2. Therefore, assuming $\mathcal{B}_1$ outputs 1 when $\mathcal{A}$ wins, we have $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_1)$.

Game_3 : This is the same as Game_2, except that the way the challenger answers opening queries is further modified. Rather than using the secret key osk to decrypt the ciphertext ct and identify the index I of the real signing key (as $\mathsf{ARS.Open}$ does in the honest protocol), the challenger instead runs the online extractor $\mathsf{OnlineExtract}$ for $\Pi_{\mathsf{NIZK,lbl}}$ to extract the witness (I, sk, r) from $(\mathsf{ct}, \pi_{\mathtt{sign}})$, and then returns the user R_I. We consider an adversary $\mathcal{B}_2$ against the online extractability of $\Pi_{\mathsf{NIZK,lbl}}$ that simulates Game_3 for $\mathcal{A}$ such that

- random-oracle queries from $\mathcal{A}$ are replied by querying $(\mathtt{hash}, \cdot)$ (see Def. 2.10);
- instead of computing $\pi_{\mathtt{sign}}$ when answering a signing query, $\mathcal{B}_2$ makes a query $(\mathtt{prove}, \mathsf{M}, \mathsf{x}, \mathsf{w})$, where $(\mathsf{x}, \mathsf{w}) = ((\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r))$, and
- instead of running $\mathsf{OnlineExtract}$, $\mathcal{B}_2$ makes a query $(\mathtt{extract}, \mathsf{M}, \mathsf{x}, \pi_{\mathtt{sign}})$.

Note that `extract` for proofs originating from `prove` queries are answered with $\perp$, which is compatible with the fact that the challenger outputs $\perp$ for opening queries that correspond to signatures originating from the signing oracle in Game_3. If $\mathcal{B}_2$ loses the multi-proof online extractability game (i.e., $\mathcal{B}_2$ did not cause the extractor to fail), then it follows from the $(\mathcal{R}', \mathcal{KR}')$-correctness of Π_{PKE} that for each extraction $\mathsf{W} = (I, \mathsf{sk}, r)$ we have $\mathsf{Dec}(\mathsf{osk}, \mathsf{ct}) = \mathsf{Dec}(\mathsf{osk}, \mathsf{Enc}(\mathsf{opk}, I; r)) = I$, so the view of $\mathcal{A}$ is not affected by whether I was obtained from $\mathsf{OnlineExtract}$ or by decrypting ct with osk. Therefore, we have $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2)$.

Game_4 : This is the same as Game_3, except that we change how the challenger answers signing queries from the adversary: The challenger generates ct as in Game_3, but uses the zero-knowledge simulator Sim for $\Pi_{\mathsf{NIZK,lbl}}$ to create the proof $\pi_{\mathtt{sign}}$ rather than using $\mathsf{NIZK.Prove}_{\mathsf{lbl}}$. It then outputs $(\mathsf{ct}, \pi_{\mathtt{sign}})$ as the signature. Similarly to the transition from Game_1 to Game_2, we can define an adversary $\mathcal{B}_3$ against the zero-knowledge property of $\Pi_{\mathsf{NIZK,lbl}}$ such that $\mathsf{Adv}_3(\mathcal{A}) \leq \mathsf{Adv}_4(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_3)$.

Game_5 : This is the same as Game_4, except we further change how the challenger answers signing queries: Instead of encrypting the correct index I to obtain ct, the challenger encrypts a random index I'. We define a multi-challenge $\mathsf{IND\text{-}CPA}$ adversary $\mathcal{B}_4$ for Π_{PKE} that simulates Game_5 for $\mathcal{A}$, but instead of generating $(\mathsf{opk}, \mathsf{osk})$, the adversary $\mathcal{B}_4$ receives opk from the multi-challenge $\mathsf{IND\text{-}CPA}$ challenger, and instead of producing the ciphertexts ct the adversary $\mathcal{B}_4$ makes encryption queries (I, I'), where I is the correct index, and I' is a random index. Note that, say on input $(\texttt{sign}, \mathsf{R}, \mathsf{M}, \mathsf{sk}_0, \mathsf{sk}_1)$, the I-th key in R is the verification key corresponding to sk_0. We can make this replacement because in Game_5, the challenger does not use osk. (The purpose of Game_2 and Game_3 were to remove the use of osk for this reason.) If the hidden bit b in the $\mathsf{IND\text{-}CPA}$ game is 0, then the $\mathsf{IND\text{-}CPA}$ experiment is identical to Game_4, and if the bit is 1, then the experiment is equal to Game_5. Therefore, we have that $\mathsf{Adv}_4(\mathcal{A}) \leq \mathsf{Adv}_5(\mathcal{A}) + \mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{B}_4)$.

Finally, observe that in Game_5 the challenger leaks no information about the secret bit b because b is not used. Hence, $\mathsf{Adv}_5(\mathcal{A}) = 0$.

□

Remark 3.5. *In the previous proof we really relied on the online extractability property (without rewinding). This is because, even if we allow for a non-tight reduction, we cannot resort to rewinding (i.e., the forking lemma) since there can be polynomially many open queries and the reduction loss will be exponential if we try to extract from all of them. Here, keep in mind that the online extractor must succeed with (roughly)* $1 - \mathsf{negl}(\lambda)$ *rather than any non-negligible function* $1/\mathsf{poly}(\lambda)$ *since there can be polynomially many open queries. Namely, even a success probability of* $1/2$ *will not be good enough. Most, if not all, prior works circumvent this issue by using an IND-CCA* PKE *as building block rather than a (possibly inefficient) online extractable* NIZK *to simulate the decryption of* ct.

Theorem 3.6. *The accountable ring signature scheme* Π_{ARS} *in Fig. 1 is unforgeable in the random oracle model. More precisely, for any adversary* $\mathcal{A}$ *that runs in time* T *and makes* $\mathcal{Q}_u$ *queries to the* `ukeygen` *oracle, there exist adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*, running in time* $O(T)$*, such that*

$$\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + \mathcal{Q}_u \mathsf{Adv}^{\mathsf{Hard}}_{\mathsf{RelSetup,IGen}}(\mathcal{B}_3)$$

Proof. We prove unforgeability using a hybrid argument with the following series of games. Let the advantage of the adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

Game_1 : This is the original unforgeability game defined in Def. 2.18. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$ by definition.

Game_2 : This is the same as Game_1, but the winning condition is changed. We let the challenger maintain a list $L_{\mathcal{O}}$ of all the random oracle queries that $\mathcal{A}$ makes. When $\mathcal{A}$ finishes the game by outputting $(\mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma = (\mathsf{ct}, \pi_{\texttt{sign}}), \pi)$, the challenger runs $(I, \mathsf{sk}, r) \leftarrow \mathsf{OnlineExtract}(\mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\texttt{sign}}, L_{\mathcal{O}})$. The game results in a loss if $((\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r)) \notin \tilde{R}_{\mathsf{sig}}$, otherwise, the winning condition is not changed. We construct an online-extractability adversary $\mathcal{B}_1$ for $\Pi_{\mathsf{NIZK,lbl}}$ that simulates Game_2 for $\mathcal{A}$. He replies random-oracle queries from $\mathcal{A}$ by querying $(\texttt{hash}, \cdot)$ (see Def. 2.10), signing queries by making an oracle call $(\texttt{prove}, \mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r))$ instead of computing $\pi_{\texttt{sign}}$ himself, and makes the oracle call $(\texttt{extract}, \mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\texttt{sign}})$ instead of running $\mathsf{OnlineExtract}$. The view of $\mathcal{A}$ during the game simulated by $\mathcal{B}_1$ is identical to its view during Game_1 and Game_2. Suppose that the output received by $\mathcal{A}$ is a win for the winning condition of Game_1, but a loss for the winning condition of Game_2. This means that $\mathsf{NIZK.Verify}^{\mathcal{O}}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\texttt{sign}}) = \top$ and $(\mathsf{ct}, \pi_{\texttt{sign}})$ was not the output of a query $(\texttt{sign}, \mathsf{opk}, \mathsf{vk}', \mathsf{R}, \mathsf{M})$ for any vk', otherwise the winning condition of Game_1 would not be met. Moreover, we would have $((\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r)) \notin \tilde{R}_{\mathsf{sig}}$, otherwise the winning condition of Game_2 would be met. This is precisely the situation $\mathcal{B}_1$ needs in order to win the online extractability game. Therefore, we have $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1)$

Game_3 : This is the same as Game_2 except that we change the way the challenger answers signing queries from $\mathcal{A}$. Specifically, the challenger generates ct as in Game_2 but uses the zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ for $\Pi_{\mathsf{NIZK,lbl}}$ to create the proof $\pi_{\texttt{sign}}$. That is, it forwards the random-oracle queries to Sim_0, and runs Sim_1 to get $\pi_{\texttt{sign}}$. It then outputs $(\mathsf{ct}, \pi_{\texttt{sign}})$ as the signature. Let $\mathcal{B}_2$ be an adversary against the zero-knowledge property of $\Pi_{\mathsf{NIZK,lbl}}$, which simulates Game_3 for $\mathcal{A}$ by forwarding random-oracle queries and proving queries to the oracles Sim_0 and Sim_1, respectively. If $\mathcal{B}_2$ is given access to oracles $\mathcal{O}$ and Prove (see Def. 2.8), then $\mathcal{A}$'s view is identical to Game_2, and if $\mathcal{B}_2$ is run with access to $\mathsf{Sim}_0, \mathsf{Sim}_1$, then $\mathcal{A}$'s view is identical to Game_3. Therefore, we have $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2)$.

Game_4 : This is the same as Game_3 except that we change the winning condition again: the challenger guesses a random index $\widetilde{I} \in \{1, \ldots, \mathcal{Q}_u\}$ at the outset of the game. If $\mathcal{A}$ makes a corruption query to corrupt the verification key returned in the $\widetilde{I}$-th user key generation query, then Game_4 aborts. The game results in a win if the winning condition of Game_3 is met and if $\widetilde{I} = I$. Since $\widetilde{I}$ is information-theoretically hidden during the execution of the game, we have $\widetilde{I} = I$ with probability $1/\mathcal{Q}_u$. Therefore, we have $\mathsf{Adv}_3(\mathcal{A}) = \mathcal{Q}_u \mathsf{Adv}_4(\mathcal{A})$.

Finally, let $\mathcal{B}_3$ be an adversary against $(\mathsf{RelSetup}, \mathsf{IGen})$ which simulates Game_4 for $\mathcal{A}$. At the beginning of the game, $\mathcal{B}_3$ is given an instance $(\mathsf{pp}_1, \mathsf{x})$. The adversary $\mathcal{B}_3$ simulates an execution of Game_4 by using the public parameter pp_1 that is given to him, rather than generating a new pp_1 himself using $\mathsf{RelSetup}$, and by answering the $\widetilde{I}$-th `ukeygen` query assigning $\mathsf{vk}_{\widetilde{I}} = \mathsf{x}$ instead of running $(\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{IGen}(\mathsf{pp}_1)$. Note that $\mathcal{B}_3$ does not need w because if $\mathcal{A}$ makes a query to corrupt $\mathsf{vk}_{\widetilde{I}}$ then the game aborts. The view of $\mathcal{A}$ during $\mathcal{B}_3$'s simulation is the same as its view during a real execution of Game_4, so $\mathsf{OnlineExtract}$ outputs a valid witness $(\widetilde{I}, \mathsf{sk}, r)$ with probability at least $\mathsf{Adv}_4(\mathcal{A})$. If this is the case, then $\mathcal{B}_3$ wins his game against the hardness of $(\mathsf{RelSetup}, \mathsf{IGen})$ by outputting sk. Therefore, we have $\mathsf{Adv}_4(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{Hard}}_{\mathsf{RelSetup,IGen}}(\mathcal{B}_3)$. ☐

Theorem 3.7. *The accountable ring signature scheme* Π_{ARS} *in Fig. 1 is traceable and tracing sound in the random oracle model. More precisely, for any adversary* $\mathcal{A}$ *that runs in time* T*, we have adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ *that run in time* $O(T)$*, such that*

$$\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1)$$

and

$$\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_3)$$

Proof. We prove the two properties separately as follows:

Traceability. Traceability follows from the statistical soundness of $\Pi_{\mathsf{NIZK,lbl}}$, the $(\mathcal{R}', \mathcal{KR}')$-correctness of Π_{PKE}, and the correctness of Π_{NIZK}. Observe that if $\mathcal{A}$ wins an execution of the traceability game, then $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, \mathsf{X} = (\mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\mathtt{sign}}) = \top$, but still there cannot be a witness $\mathsf{W} = (I, \mathsf{sk}, r)$ such that $(\mathsf{X}, \mathsf{W}) \in \tilde{R}_{\mathsf{sig}}$. Towards a contradiction, suppose that such a witness does exist, then the $(\mathcal{R}', \mathcal{KR}')$-correctness of the PKE implies that $\mathsf{Dec}(\mathsf{osk}, \mathsf{ct} = \mathsf{Enc}(\mathsf{opk}, I; r)) = I$, which implies that $((\mathsf{opk}, \mathsf{ct}, I), \mathsf{osk}) \in R_{\mathsf{open}}$, so the correctness of Π_{NIZK} implies that $\mathsf{NIZK.Verify}((\mathsf{opk}, \mathsf{ct}, I), \pi_{\mathtt{open}}) = \top$. This means that $\mathcal{A}$ did not win the traceability game. Therefore, $\mathcal{A}$ produces valid proofs for statements not in $\tilde{R}_{\mathsf{sig}}$ with probability at least $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$. We can use this to construct an adversary $\mathcal{B}_1$ against the statistical soundness of $\Pi_{\mathsf{NIZK,lbl}}$ that generates $\mathsf{pp} \leftarrow \mathsf{ARS.Setup}(1^\lambda)$ for a security parameter λ, runs $(\mathsf{rr}, \mathsf{R}, \mathsf{M}, \sigma) \leftarrow \mathcal{A}(\mathsf{pp})$ where $\sigma = (\mathsf{ct}, \pi_{\mathtt{sign}})$, and $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{ARS.OKGen}(\mathsf{pp}; \mathsf{rr})$, and outputs $(\mathsf{M}, \mathsf{x} := (R, \mathsf{opk}, \mathsf{ct}), \pi_{\mathtt{sign}})$, which makes $\mathcal{B}_1$ win. B_1's advantage is therefore $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1)$.

Tracing soundness. Similarly, tracing soundness follows from the statistical soundness of Π_{NIZK} and $\Pi_{\mathsf{NIZK,lbl}}$, and the $(\mathcal{R}', \mathcal{KR}')$-correctness of the Π_{PKE}. In order for $\mathcal{A}$ to win the tracing soundness game, it needs to output valid proofs $\pi_{\mathtt{sign}}$, π_0, π_1 (the former is part of the produced signature $\sigma = (\mathsf{ct}, \pi_{\mathtt{sign}})$) such that there exist witnesses (I, sk, r), osk_0 and osk_1 where

$$\begin{aligned} ((\mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, \mathsf{sk}, r)) &\in \tilde{R}_{\mathsf{sig}} \\ ((\mathsf{opk}, \mathsf{ct}, I_0), \mathsf{osk}_0) &\in \tilde{R}_{\mathsf{open}} \\ ((\mathsf{opk}, \mathsf{ct}, I_1), \mathsf{osk}_1) &\in \tilde{R}_{\mathsf{open}}, \end{aligned}$$

with $I_0 \neq I_1$. However, it follows from the $(\mathcal{R}', \mathcal{KR}')$-correctness of Π_{PKE} that no three such witnesses can exist. Suppose, towards a contradiction, that those witnesses exist. Then we have $I_0 = \mathsf{Dec}(\mathsf{osk}_0, \mathsf{ct} = \mathsf{Enc}(\mathsf{opk}, I; r))$, so the $(\mathcal{R}', \mathcal{KR}')$-correctness implies that $I_0 = I$, and similarly it follows from $I_1 = \mathsf{Dec}(\mathsf{osk}_1, \mathsf{ct} = \mathsf{Enc}(\mathsf{opk}, I; r))$ that $I_1 = I$, which contradicts $I_0 \neq I_1$. Therefore, at least one of $\pi_{\mathtt{sign}}, \pi_0, \pi_1$ is a valid proof of an invalid statement, i.e. a X for which does not exist W such that $(\mathsf{X}, \mathsf{W}) \in \tilde{R}_{\mathsf{sig}}$ (or $(v) \in \tilde{R}_{\mathsf{open}}$), with probability at least $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$. Let $\mathcal{B}_2$ and $\mathcal{B}_3$ be statistical-soundness adversaries for $\Pi_{\mathsf{NIZK,lbl}}$ and Π_{NIZK}, respectively, that simulate the tracing soundness game and output $\pi_{\mathtt{sign}}$ or π_b, respectively, where b is a random bit. Then we have $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_3)$. $\square$

3.2 Accountable Ring Signature to Dynamic Group Signature

Accountable ring signatures are known to trivially imply dynamic group signatures [BCC+15, BCC+16]. A formal treatment is provided by Bootle et al. [BCC+16]. We remark that the transformation provided in [BCC+16] retains the the same level of security provided by the underlying accountable ring signature. That is, all reductions between unforgeability, full-anonymity and traceability are tight. For completeness, we provide more details on group signatures and the transform in App. B.

3.3 Tightly Secure Variant

Observe the only source of loose reduction in the previous section was in the unforgeability proof (see Thm. 3.6), where we assume each building blocks, i.e., NIZK and PKE, are tightly reduced to concrete hardness assumptions. In this subsection, we apply the Katz-Wang technique [KW03] to modify our construction in Fig. 1 to obtain a tight reduction.

We firstly give an intuition of the method. Recall that in the proof of Thm. 3.6, the reduction is given a challenge instance x, guesses which user's signature the adversary will forge, and assigns x to the verification key vk of the selected user. If the adversary queries the corruption oracle on the key vk, the reduction fails and aborts since it will not be able to produce the corresponding secret key for vk. If the guess is correct and the adversary successfully forges the signature, then the reduction can recover a witness w' such that $(\mathsf{x}, \mathsf{w}')$ is in the relation $\widetilde{R}_{\mathsf{pp}_1}$. Therefore, if the adversary makes $\mathcal{Q}_u$ user key generation queries and its advantage is ϵ, then the reduction can extract a witness with probability roughly $\epsilon/\mathcal{Q}_u$.

A high-level viewpoint of the Katz-Wang method is that each user is given a pair of statements $(\mathsf{x}^{(1)}, \mathsf{x}^{(2)})$ as the verification key vk, with only one witness w as the secret signing key, such that either $(\mathsf{x}^{(1)}, \mathsf{w})$ or $(\mathsf{x}^{(2)}, \mathsf{w})$ is in the relation $\widetilde{R}_{\mathsf{pp}_1}$. Also, we assume that now the reduction is given $\mathcal{Q}_u$ challenge instances $\{\mathsf{x}_i\}_{i\in[\mathcal{Q}_u]}$ and it is required to solve any one of them. The reduction in this case needs no guessing steps as above. Specifically, the reduction can use IGen to generate pairs $(\widetilde{\mathsf{x}}_i, \widetilde{\mathsf{w}}_i)$ for $i \in [\mathcal{Q}_u]$, randomly permutes $\mathsf{x}_i, \widetilde{\mathsf{x}}_i$ and assigns the obtained ordered pair to vk_i. Therefore, the reduction can always answer any corruption query with $\widetilde{\mathsf{w}}_i$. As long as the adversary wins the unforgeability game by forging a signature, the reduction can return a witness for one of the $\{\mathsf{x}_i\}_{i\in[\mathcal{Q}_u]}$ with probability 1/2. Roughly speaking, if the success rate of the adversary is ϵ, then the reduction can extract the answer for the challenge $(\star, X_0, \{\mathsf{x}_i\}_{i\in[\mathcal{Q}_u]})$ with probability around $\epsilon/2$. Here, it is important that the information on which verification key the user knows the signing key to needs to remain hidden from the adversary. Otherwise, the adversary may always create a forgery with respect to the signing key the reduction already knows.

To turn the above idea into a formal proof, we require two new ingredients: an instance generator that outputs multiple challenges and a NIZK that additionally hides the information on which signing key is used. More formally, we build a tightly secure accountable ring signature scheme $\Pi^{\mathsf{Tight}}_{\mathsf{ARS}} = (\mathsf{ARS.Setup}, \mathsf{ARS.OKGen}, \mathsf{ARS.UKGen}, \mathsf{ARS.Sign}, \mathsf{ARS.Verify}, \mathsf{ARS.Open}, \mathsf{ARS.Judge})$ based on the following tools. The only difference between the tools used in Sec. 3.1 are the hard multi-instance generator and the NIZK for the relation $R^{\mathsf{Tight}}_{\mathsf{sig}}$.

- A hard *multi-instance* generator $(\mathsf{RelSetup}, \mathsf{IGen})$ contains a setup algorithm $\mathsf{RelSetup}$ that outputs a description pp of a pair of relations $R_{\mathsf{pp}} \subseteq \tilde{R}_{\mathsf{pp}}$, and an instance generator IGen for these pairs of relations. That is, $\mathsf{RelSetup}$ and IGen are PPT algorithms such that $\Pr[(\mathsf{x}_i, \mathsf{w}_i) \in R_{\mathsf{pp}} \mid \mathsf{pp} \leftarrow \mathsf{RelSetup}(1^\lambda); \{(\mathsf{x}_i, \mathsf{w}_i)\}_{i \in [N]} \leftarrow \mathsf{IGen}(\mathsf{pp}, N)] = 1$. Moreover, if we define the advantage of an adversary $\mathcal{A}$ against $(\mathsf{RelSetup}, \mathsf{IGen})$ as

$$\mathsf{Adv}^{\mathsf{Multi\text{-}Hard}}_{\mathsf{RelSetup},\mathsf{IGen},N}(\mathcal{A}) = \Pr\left[(\mathsf{x}_i, \mathsf{w}') \in \tilde{R}_{\mathsf{pp}} \,\middle|\, \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{RelSetup}(1^\lambda) \\ \{(\mathsf{x}_i, \mathsf{w}_i)\}_{i \in [N]} \leftarrow \mathsf{IGen}(\mathsf{pp}, N) \\ (i, \mathsf{w}') \leftarrow \mathcal{A}(\mathsf{pp}, \{\mathsf{x}_i\}_{i \in [N]}) \end{array}\right]$$

then $\mathsf{Adv}^{\mathsf{Multi\text{-}Hard}}_{\mathsf{RelSetup},\mathsf{IGen},N}(\mathcal{A})$ is a negligible function in λ for every PPT adversary $\mathcal{A}$.

- A public-key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with multi-challenge IND-CPA security, and with $(\mathcal{R}', \mathcal{KR}')$-correctness for some relaxed randomness set $\mathcal{R}'$ and some relaxed key relation $\mathcal{KR}'$. The message space of the encryption scheme contains a set of indices $[N]$ for any polynomially large $N \in \mathbb{N}$.
- A multi-proof online extractable NIZK proof system with labels $\Pi_{\mathsf{NIZK},\mathsf{lbl}} = (\mathsf{NIZK.Setup}_{\mathsf{lbl}}, \mathsf{NIZK.Prove}_{\mathsf{lbl}}, \mathsf{NIZK.Verify}_{\mathsf{lbl}})$ for the family of relations

$$R^{\mathsf{Tight}}_{\mathsf{sig}} = \left\{ \left((\mathsf{pp}, \{\mathsf{x}^{(j)}_i\}_{(i,j) \in [N] \times [2]}, \mathsf{pk}, \mathsf{ct}), (I, b, \mathsf{w}, r) \right) \,\middle|\, \begin{array}{c} (I, r) \in [N] \times \mathcal{R} \wedge (\mathsf{x}^{(b)}_I, \mathsf{w}) \in R_{\mathsf{pp}} \wedge \\ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}$$
$$\tilde{R}^{\mathsf{Tight}}_{\mathsf{sig}} = \left\{ \left((\mathsf{pp}, \{\mathsf{x}^{(j)}_i\}_{(i,j) \in [N] \times [2]}, \mathsf{pk}, \mathsf{ct}), (I, b, \mathsf{w}, r) \right) \,\middle|\, \begin{array}{c} (I, r) \in [N] \times \mathcal{R}' \wedge (\mathsf{x}^{(b)}_I, \mathsf{w}) \in \tilde{R}_{\mathsf{pp}} \wedge \\ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\} .$$

- A second NIZK proof system (without labels) $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the family of relations

$$R_{\mathsf{open}} = \{((\mathsf{pk}, \mathsf{ct}, I), \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR} \wedge \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = I\}$$
$$\tilde{R}_{\mathsf{open}} = \left\{((\mathsf{pk}, \mathsf{ct}, I), \mathsf{sk}) \,\middle|\, (\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR}' \wedge \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = I\right\} ,$$

with statistical soundness (Def. 2.9).

The building blocks listed above are combined similarly to Fig. 1. For the sake of completeness, we detail the resulting protocol in Fig. 2. For the security properties, we only focus on unforgeability. The others are a direct consequence of the proofs given for the non-tight construction in Fig. 1.

Theorem 3.8. *The accountable ring signature scheme $\Pi^{\mathsf{Tight}}_{\mathsf{ARS}}$ in Fig. 2 is unforgeable in the random oracle model. More precisely, for any adversary $\mathcal{A}$ that runs in time T and makes $\mathcal{Q}_u$ queries to the* `ukeygen` *oracle, there exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, running in time $O(T)$, such that*

$$\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK},\mathsf{lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK},\mathsf{lbl}}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{Multi\text{-}Hard}}_{\mathsf{RelSetup},\mathsf{IGen},\mathcal{Q}_u}(\mathcal{B}_3).$$

Proof. We prove unforgeability using a hybrid argument with the following series of games. Let the advantage of an adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

- The first game, Game_1, is the original unforgeability game defined in Def. 2.18. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{ARS}}(\mathcal{A})$ by definition.
- Game_2 is the same as Game_1, but with a modified winning condition. We let the challenger maintain a list $L_{\mathcal{O}}$ of all the random-oracle queries that $\mathcal{A}$ makes. When $\mathcal{A}$ finishes the game by outputting $(\mathsf{opk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma = (\mathsf{ct}, \pi_{\mathtt{sign}}), \pi)$, the challenger runs $(I, b, \mathsf{sk}, r) \leftarrow \mathsf{OnlineExtract}(\mathsf{M}, (\mathsf{pp}_1, \mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\mathtt{sign}}, L_{\mathcal{O}})$. The game results in a loss if $((\mathsf{pp}_1, \mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, , b, \mathsf{sk}, r)) \notin \tilde{R}^{\mathsf{Tight}}_{\mathsf{sig}}$, otherwise, the winning condition is not changed. As we have shown in the proof of Thm. 3.6, there exists an online-extractability adversary $\mathcal{B}_1$ for $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ running in time $O(T)$ such that $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK},\mathsf{lbl}}}(\mathcal{B}_1)$.

- The third game, Game_3, is the same as Game_2 except that we change the way the challenger answers signing queries from $\mathcal{A}$. Specifically, the challenger generates ct as in Game_2 but uses the $\Pi_{\mathsf{NIZK,lbl}}$ zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ to create the proof $\pi_{\mathtt{sign}}$. As we have shown in the proof of Thm. 3.6, there exists a zero-knowledge adversary $\mathcal{B}_2$ for $\Pi_{\mathsf{NIZK,lbl}}$ running in time $O(T)$ and such that $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2)$.

- Finally, we consider an adversary $\mathcal{B}_3$ against the hardness of $(\mathsf{RelSetup}, \mathsf{IGen})$ which simulates Game_3 for $\mathcal{A}$. At the beginning of the game, the adversary $\mathcal{B}_3$ is given the instances $(\mathsf{pp}_1, \{\mathsf{x}\}_{i\in[\mathcal{Q}_u]})$. $\mathcal{B}_3$ uses the public parameter pp_1 that is given to him, rather than generating new pp_1 himself using $\mathsf{RelSetup}$. Moreover, when answering the i-th `ukeygen` query, $\mathcal{B}_3$ uniformly draws b_i from $\{1, 2\}$, generates $(\widetilde{\mathsf{x}}_i, \widetilde{\mathsf{w}}_i) \leftarrow \mathsf{IGen}(\mathsf{pp}_1)$, and assigns $\mathsf{vk}_i = (\mathsf{x}_i^{(1)}, \mathsf{x}_i^{(2)})$ where $(\mathsf{x}_i^{(b_i)}, \mathsf{x}_i^{(3-b_i)}) = (\widetilde{\mathsf{x}}_i, \mathsf{x}_i)$. Note that now $\mathcal{B}_3$ is able to respond to any valid corruption query `corrupt`. In fact, for any $i \in [\mathcal{Q}_u]$, if $\mathcal{A}$ makes a corruption query to corrupt vk_i, then $\mathcal{B}_3$ responds by $\mathsf{sk} = (b_i, \widetilde{\mathsf{w}}_i)$. The view of $\mathcal{A}$ during $\mathcal{B}_3$'s simulation is the same as its view during a real execution of Game_3, so $\mathsf{OnlineExtract}$ outputs a valid witness $(\widetilde{I}, \mathsf{sk} = (b', \mathsf{w}'), r)$ with probability at least $\mathsf{Adv}_3(\mathcal{A})$. Since the sampling of the statements and witnesses follows the same distribution determined by $\mathsf{IGen}(\mathsf{pp}_1)$ in the real execution, there is an 1/2 chance that $b' = (3 - b_{\widetilde{I}})$. That is, $(\mathsf{x}_{\widetilde{I}}, \mathsf{w}') \in \widetilde{R}_{\mathsf{pp}_1}$. Therefore, we have $\mathsf{Adv}_3(\mathcal{A})/2 \leq \mathsf{Adv}^{\mathsf{Multi\text{-}Hard}}_{\mathsf{RelSetup,IGen},\mathcal{Q}_u}(\mathcal{B}_3)$.

$\square$

```
ARS.Setup($1^\lambda$)
$\mathsf{pp}_1 \leftarrow \mathsf{RelSetup}(1^\lambda)$
$\mathsf{pp}_2 \leftarrow \mathsf{PKE.Setup}(1^\lambda)$
return $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$
```

```
ARS.UKGen(pp)
$(\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathsf{pp}$
$b \xleftarrow{\$} \{1, 2\}$
$(\mathsf{x}^{(1)}, \mathsf{w}^{(1)}), (\mathsf{x}^{(2)}, \mathsf{w}^{(2)}) \leftarrow \mathsf{IGen}(\mathsf{pp}_1)$
return $(\mathsf{vk} := (X^{(1)}, X^{(2)}), \mathsf{sk} := (b, \mathsf{w}^{(b)})$
```

```
ARS.Verify(opk, R, M, $\sigma$)
$(\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathsf{pp}$
$(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
return $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{pp}_1, \mathsf{R}, \mathsf{opk}, \mathsf{ct}), \pi_{\mathtt{sign}})$
```

```
ARS.Judge(opk, R, vk, M, $\sigma$, $\pi$)
$(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
if $\nexists I : \mathsf{vk} = \mathsf{R}_I$ then
    return $\bot$.
$b_0 \leftarrow \mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma)$
$b_1 \leftarrow \mathsf{NIZK.Verify}((\mathsf{opk}, \mathsf{ct}, I), \pi_{\mathtt{open}})$
return $b_0 \wedge b_1$
```

```
ARS.OKGen(pp)
$(\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathsf{pp}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_2)$
return $(\mathsf{opk} := \mathsf{pk}, \mathsf{osk} := \mathsf{sk})$
```

```
ARS.Sign(opk, sk, R, M)
$(\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathsf{pp}$
$\{(\mathsf{x}_i^{(1)}, \mathsf{x}_i^{(2)})\}_{i\in[N]} \leftarrow \mathsf{R}$
if $\nexists (I, b) : (\mathsf{x}_I^{(b)}, \mathsf{sk}) \in R_{\mathsf{pp}_1}$ then
    return $\bot$.
$r \xleftarrow{\$} \mathcal{R}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{opk}, I; r)$
$\pi_{\mathtt{sign}} \leftarrow \mathsf{NIZK.Prove}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{pp}_1, \mathsf{R}, \mathsf{opk}, \mathsf{ct}), (I, b, \mathsf{sk}, r))$
return $\sigma := (\mathsf{ct}, \pi_{\mathtt{sign}})$
```

```
ARS.Open(osk, R, M, $\sigma$)
if $\mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \bot$ then
    return $\bot$
$(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
$I \leftarrow \mathsf{Dec}(\mathsf{osk}, \mathsf{ct})$
$\pi_{\mathtt{open}} \leftarrow \mathsf{NIZK.Prove}((\mathsf{opk}, \mathsf{ct}, I), \mathsf{osk})$
return $\pi := (\mathsf{R}_I, \pi_{\mathtt{open}})$
```

Figure 2: Modified tightly-secure construction of an accountable ring signature $\Pi^{\mathsf{Tight}}_{\mathsf{ARS}}$ obtained from a hard multi-instance generator $(\mathsf{RelSetup}, \mathsf{IGen})$, a public-key encryption algorithm $(\mathsf{PKE.Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfying some suitable correctness and security properties, a NIZK proof system with labels $\Pi_{\mathsf{NIZK,lbl}}$ for $R^{\mathsf{Tight}}_{\mathsf{sig}}$, and a NIZK proof system without labels Π_{NIZK} for R_{open}.

4 Group-Action-Based Hard Instance generators and PKEs

In this section, we introduce group-action-based hard instance generators (HIGs) and group-action-based PKEs. These are classes of HIGs and PKEs, that derive their security from cryptographic group actions, and which have some specific internal structure. We define these concepts because, as we will see in Sections 5 and 6, if we instantiate our generic accountable ring signature construction with a group-action-based HIG and a group-action-based PKE, then we can construct a very efficient multi-proof online extractable NIZK for the R_{sig} relation. We provide concrete instantiations of group-action-based HIGs and PKEs from lattices and isogenies in Sec. 7.

4.1 Group-Action-based Hard Instance Generator

We consider a special class of hard instance generators naturally induced by cryptographic hard actions.

Definition 4.1 (Group-Action-based Hard Instance Generator)**.** *A group-action-based hard instance generator,* GA-HIG *in short, is a pair of efficient algorithms* (RelSetup, IGen) *with the following properties:*

- *On input a security parameter* λ*,* RelSetup *outputs* $\mathsf{pp} = (G, S_1, S_2, \delta, X_0, \mathcal{X}, \star)$ *such that:* G *is an additive group whose elements can be represented uniquely,* $S_1 \subseteq S_2$ *are symmetric subsets of* G*, such that membership in* S_1 *and* S_2 *can be decided efficiently, and such that the group law can be computed efficiently for elements in* $S_1 \cup S_2$*. Moreover the intersection* $S_3 = \cap_{g \in S_1} g + S_2$ *has cardinality* $\delta\,|S_2|$ *and membership of* S_3 *can be decided efficiently.* $\star$ *is an action* $\star : G \times \mathcal{X} \to \mathcal{X}$ *of* G *on a set* $\mathcal{X}$ *that contains the element* X_0*.* $\star$ *can be evaluated efficiently on elements of* $S_1 \cup S_2$*. These parameters describe an NP-relation*
$$R_{\mathsf{pp}} = \{(X, s) \,|\, s \in S_1 : s \star X_0 = X\}\,,$$
and a relaxed NP-relation
$$\tilde{R}_{\mathsf{pp}} = \{(X, s) \,|\, s \in S_2 + S_3 : s \star X_0 = X\}\,.$$
- *On input* pp*,* IGen *samples an element* s *from* S_1 *and outputs* $(s \star X_0, s) \in R_{\mathsf{pp}}$*.*
- (RelSetup, IGen) *is a hard instance generator as defined in Sec. 3.*

4.2 Group-Action-based PKE

We also consider group actions provided with a corresponding public-key encryption scheme, as specified in the following definition.

Definition 4.2 (Group-action-based PKE)**.** *A group-action-based public-key encryption scheme,* GA-PKE *in short, is a public-key encryption scheme* $\Pi_{\mathsf{GA\text{-}PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *On input a security parameter* 1^λ*, it returns the public parameter* $\mathsf{pp} = (G, G_{\mathsf{M}}, \mathcal{X}, S_1, S_2, \delta, D_{\mathcal{X}}, \star_{\mathsf{M}}, \mathcal{M})$ *(sometimes implicitly) used by the scheme. Here,* G, G_{M} *are additive groups,* S_1, S_2 *two symmetric subsets of* G*,* $\mathcal{X}$ *a finite set,* δ *a real number in* $[0, 1]$*,* $D_{\mathcal{X}}$ *a distribution over a set of group actions* $\star_{\mathsf{pk}} : G \times \mathcal{X} \to \mathcal{X}$ *and elements in* $\mathcal{X}$*,* $\star_{\mathsf{M}} : G_{\mathsf{M}} \times \mathcal{X} \to \mathcal{X}$ *a group action,* $\mathcal{M} \subseteq G_{\mathsf{M}}$ *a message space. For any polynomially large* $N \in \mathbb{N}$*, we assume that there exists a feasible and invertible embedding* τ *from the set of index* $[N]$ *into the message space* $\mathcal{M}$*. For simplicity, we will write* $\tau(i) \star_{\mathcal{M}} X$*,* $\mathsf{Enc}(\mathsf{pk}, \tau(i))$ *as* $i \star_{\mathsf{M}} X$*,* $\mathsf{Enc}(\mathsf{pk}, i)$ *respectively without causing confusion.*

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: *On input a public parameter* pp*, it returns a public key* pk *and a secret key* sk*. We assume* $\mathsf{pk} = (\star_{\mathsf{pk}}, X_{\mathsf{pk}})$ *to be drawn from* $D_{\mathcal{X}}$*, where* $\star_{\mathsf{pk}} : G \times \mathcal{X} \to \mathcal{X}$ *is a group action and* $X_{\mathsf{pk}} \in \mathcal{X}$*, and* $\mathsf{sk} \in G$*. We also assume* pk *includes* pp *w.l.o.g.*

$\mathsf{Enc}(\mathsf{pk}, \mathsf{M}; r) \to \mathsf{ct}$: *On input a public key* $\mathsf{pk} = (\star_{\mathsf{pk}}, X_{\mathsf{pk}})$ *and a message* $\mathsf{M} \in \mathcal{M}$*, it returns a ciphertext* ct*. We assume* ct *is generated as* $\mathsf{M} \star_{\mathsf{M}} (r \star_{\mathsf{pk}} X_{\mathsf{pk}}) \in \mathcal{X}$*, where the encryption randomness is sampled as* $r \xleftarrow{\$} S_1$*.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$: *On input a secret key* sk*and a ciphertext* ct*, it (deterministically) returns a message* $\mathsf{M} \in \mathcal{M}$*.*

In addition, we assume the following properties hold for the group actions defined by pp*.*

1. *There exists a positive-valued polynomial* T *such that for all* $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, and* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(\mathsf{pp})$*, one can efficiently compute* $g \star_{\mathsf{pk}} X$ *for all* $g \in S_1 \cup S_2$ *and all* $X \in \mathcal{X}$ *in time at most* $T(\lambda)$*, sample uniformly from* S_1 *and* S_2*, and represent elements of* G *and* $\mathcal{X}$ *uniquely. It is also efficient to compute the action* $\star_{\mathsf{M}}$ *for every possible input.*
2. *The intersection* S_3 *of the sets* $S_2 + g$*, with* g *varying in* S_1*, is such that its cardinality is equal to* $\delta\,|S_2|$*. Furthermore, it is efficient to check whether an element* $g \in G$ *belongs to* S_3*.*

We further require a group-action-based PKE to satisfy standard correctness and decryption efficiency.

Definition 4.3 (Correctness and Decryption Efficiency)**.** *We say a group-action-based* PKE $\Pi_{\mathsf{GA\text{-}PKE}}$ *is* correct *if for all* $\lambda \in \mathbb{N}$*, and for all but a negligible fraction of* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, we have* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{M})) = \mathsf{M}$ *for all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(\mathsf{pp})$*and* $\mathsf{M} \in \mathcal{M}$*.*

Moreover, we require Dec *to run in* $\mathsf{poly}(\lambda)$ *for a fixed polynomial function* poly *and for all possible inputs.*

As we shown in Sec. 3.1, in order to construct an accountable ring signature, a group-action-based PKE is also required to be (multi-challenge) $\mathsf{IND\text{-}CPA}$ secure (Def. 2.14) and $(\mathcal{R}', \mathcal{KR}')$-correct for some relaxed randomness set $\mathcal{R}'$ and some relaxed key relation $\mathcal{KR}'$ (Def. 3.1).

The concrete choice of $(\mathcal{R}', \mathcal{KR}')$ may depend on the instantiation. For instance, while we define $(\mathcal{R}', \mathcal{KR}') = (\mathcal{R}, \mathcal{KR})$ for our isogeny-based instantiation in Sec. 7.1, we must rely on a strictly wider relation for our lattice-based instantiation in Sec. 7.2 to compensate for the *relaxed* soundness. In slightly more detail, in our lattice-based NIZK, we are only able to argue that an adversary created a ciphertext ct using message M and randomness $r \in \mathcal{R}'$, and/or that a ct can be decrypted to M using secret key sk such that $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{KR}'$. Roughly, $(\mathcal{R}', \mathcal{KR}')$-correctness guarantees that such an argument suffices to prove that ct can only be decrypted to a unique M. Recall this argument is explicitly used in the proof of Thm. 3.7.

5 Sigma Protocol for a "Traceable" OR Relation

In this section, we present an efficient sigma protocol for the relation R_{sig} introduced in Sec. 3.1, using group-action-based HIG and a group-action-based PKE from the previous section. Recall this relation was used to define the multi-proof online extractable NIZK with labels Π_{NIZK}, which allowed an OR proof along with a proof of opening to a ciphertext. Looking ahead, in Sec. 6, we show that our sigma protocol can be turned into a multi-proof online extractable NIZK using the Fiat-Shamir transform. This is in contrast to the common application of Fiat-Shamir transform that only provides a proof of knowledge via the rewinding argument [FS87, PS00]. We note that we do not focus on the other NIZK for the relation R_{open} in Sec. 3.1 since they can be obtained easily from prior works.

We call the sigma protocol we present in this section as a *traceable* OR sigma protocol since it allows to trace the prover. This section is structured as follows. Firstly, we introduce a *base* traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ for the relation R_{sig} with proof size $O(\log N)$ but with a binary challenge space. Secondly, we amplify the soundness of the sigma protocol by performing parallel repetitions. Here, instead of applying λ-parallel repetitions naively, we optimize it using three approaches developed in [BKP20] to obtain our *main* traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$. Finally, we show a sigma protocol for the "tight" relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$ introduced in Sec. 3.3.

5.1 From a Group-Action-Based HIG and PKE to Base Traceable OR Sigma Protocol

In this section, we present a *base* OR sigma protocol for the relation R_{sig} with a binary challenge space from which the main OR sigma protocol will be deduced.

Parameters and Binary Relation. The sigma protocol is based on a group-action-based HIG and PKE. Let $\mathsf{pp}_1 = (G, \mathcal{X}, S_1, S_2, \delta_x, \star, X_0)$ and $\mathsf{pp}_2 = (\overline{G}, \overline{G}_{\mathsf{T}}, \mathcal{Y}, \overline{S}_1, \overline{S}_2, \delta_y, D_{\mathcal{Y}}, \star_{\mathsf{M}}, \mathcal{M})$ be public parameters in the image of $\mathsf{RelSetup}$ and $\mathsf{PKE.Setup}$, respectively. Moreover, let $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(\mathsf{pp}_2)$. The relation R_{sig} in Sec. 3.1 can be equivalently rewritten as follows:

$$R_{\mathsf{sig}} = \left\{ ((\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s, r)) \;\middle|\; \begin{array}{c} (I, s, r) \in [N] \times S_1 \times \overline{S}_1 \wedge \\ X_I = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}.$$

Recall that by definition of $\mathsf{GA\text{-}PKE}$ (Def. 4.2), the ciphertext ct is restricted to the simple form $I \star_{\mathsf{M}} (r \star_{\mathsf{pk}} Y_{\mathsf{pk}}) \in \mathcal{Y}$, where $r \in \overline{S}_1 \subseteq \overline{G}$.

Sigma Protocol for R_{sig}. We now sketch the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$. A prover with witness $(I, s, r) \in [N] \times S_1 \times \overline{S}_1$ first samples $(s', r') \xleftarrow{\$} S_2 \times \overline{S}_2$, and $(\{\mathsf{bits}_i\}_{i\in[N]}) \leftarrow \{0,1\}^{\lambda N}$. Then, it computes commitments

$$\mathsf{C}_i = \mathcal{O}(\mathsf{Com} \parallel s' \star X_i \parallel r' \star_{\mathsf{pk}} (-i \star_{\mathsf{M}} \mathsf{ct}) \parallel \mathsf{bits}_i) \quad \forall i \in [N],$$

and builds a Merkle tree (see App. A.2) with $(\mathsf{C}_1, \ldots, \mathsf{C}_N)$ as its leaves, obtaining root. Here, notice $r' \star_{\mathsf{pk}} (-i \star_{\mathsf{M}} \mathsf{ct}) = r' \star_{\mathsf{pk}} (-i + I) \star_{\mathsf{M}} (r \star_{\mathsf{pk}} Y_{\mathsf{pk}})$ is simply $(r' + r) \star_{\mathsf{pk}} Y_{\mathsf{pk}}$ when $i = I$. Then, the prover sends $\mathsf{com} = \mathsf{root}$ to the verifier as the commitment of the sigma protocol. The verifier, in turn, responds with a uniform challenge $\mathsf{chall} \in \{0, 1\}$.

If the challenge bit chall is 0, then the prover sends (s', r') and the commitment randomness $\{\mathsf{bits}_i\}_{i\in[N]}$. That is, all the randomness it generated in the first round. The verifier then can reconstruct the Merkle tree and verify that the root of the obtained tree is equal to root.

If the challenge bit chall is equal to 1, then the prover computes $s'' = s' + s$, $r'' = r' + r$. The prover aborts the protocol if $s'' \notin S_3$ or $r'' \notin \overline{S}_3$. The first event will occur with probability $(1 - \delta_x)$ and, similarly, the second event will occur with probability $(1 - \delta_y)$. Otherwise, the prover sends (r'', s'') together with the path connecting C_I to root in the Merkle tree, and the corresponding commitment randomness bits_I to the verifier. The verifier computes $\widetilde{\mathsf{C}}_I = \mathcal{O}(\mathsf{Com} \parallel s'' \star X_0 \parallel r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}} \parallel \mathsf{bits}_I)$ and uses the received path to reconstruct $\widetilde{\mathsf{root}}$ of the Merkle tree. The verifier checks whether $\widetilde{\mathsf{root}} = \mathsf{root}$.

To reduce the communication cost, a pseudorandom number generator (PRG) Expand can be run over a uniform seed $\mathsf{seed} \in \{0,1\}^{\lambda}$ to produce the group elements s', r' and all commitment randomness values $\mathsf{bits}_1, \ldots, \mathsf{bits}_N$ (part of the response for $\mathsf{chall} = 0$). As a consequence, if the challenge bit is 0, the prover responds with seed so that the verifier can generate $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)$ with the PRG Expand. The response corresponding to the challenge bit $\mathsf{chall} = 1$ remains unchanged. We instantiate the PRG by a random oracle $\mathcal{O}(\mathsf{Expand} \parallel \cdot)$. Looking ahead, using a PRG not only provides efficiency but it proves to be essential when proving multi-proof online extractability when compiled into a NIZK. Roughly, the seed binds the cheating prover from using arbitrary $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)$ and the random oracle allows for efficient extraction. Finally, we instantiate the collision-resistant hash function $\mathcal{H}_{\mathsf{Coll}}(\cdot)$ used in our Merkle tree by a random oracle $\mathcal{O}(\mathsf{Coll} \parallel \cdot)$.

A formal description of $\Pi_{\Sigma}^{\mathsf{base}}$ is provided in Fig. 3.

Security of Sigma Protocol $\Pi_{\Sigma}^{\mathsf{base}}$.

The following Thms. 5.1 and 5.2 summarize the security of our sigma protocol. We point out that in Thm. 5.1, we show our sigma protocol satisfies special soundness for the relations R_{sig} and $\tilde{R}'_{\mathsf{sig}}$ such that $R_{\mathsf{sig}} \subset \tilde{R}'_{\mathsf{sig}}$, rather than for the relations R_{sig} and $\tilde{R}_{\mathsf{sig}}$ such that $R_{\mathsf{sig}} \subseteq \tilde{R}_{\mathsf{sig}}$, where $\tilde{R}_{\mathsf{sig}}$ is the relaxed relation introduced in Sec. 3.1. The subtle difference is that $\tilde{R}'_{\mathsf{sig}}$ captures the scenario where the extractor may extract a witness that forms a collision in the random oracle. This has no concrete impact as we are able to turn such a sigma protocol into a multi-proof online extractable NIZK for the relations R_{sig} and $\tilde{R}_{\mathsf{sig}}$.

Theorem 5.1. *The sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$ has correctness with abort rate $(1-\delta_x\delta_y)/2$ and relaxed special soundness for the relations R_{sig} and $\tilde{R}'_{\mathsf{sig}}$, where*

$$\tilde{R}'_{\mathsf{sig}} = \left\{ \Big((\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W} \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = (I,s,r) \in [N] \times (S_2+S_3) \times (\overline{S}_2+\overline{S}_3) \\ \wedge\ X_I = s\star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \textit{ or} \\ \mathsf{W} = (x_1,x_2)\in\{0,1\}^* \wedge x_1\neq x_2 \wedge \mathcal{O}(x_1)=\mathcal{O}(x_2)\end{array} \right\}.$$

Here, $\tilde{R}'_{\mathsf{sig}}$ is identical to the one defined in Sec. 3.1 if we ignore the hash collision $\mathsf{W} = (x_1, x_2)$ *and set $\mathcal{R}' = \overline{S}_2 + \overline{S}_3$ in the $(\mathcal{R}', \mathcal{KR}')$-correctness of* GA-PKE.

Proof. <u>Correctness</u>. Say the prover honestly runs $\Pi_{\Sigma}^{\mathsf{base}}$ on an input (I, s, r) satisfying $X_I = s \star X_0$ and $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r)$, and does not abort. If $\mathsf{chall} = 0$, then the verifier repeats the computation in the commitment phase (see Round 1 in Fig. 3) and therefore obtains the same output. If $\mathsf{chall} = 1$, then the verifier computes $\widetilde{T} = s'' \star X_0$ and $\widetilde{\mathsf{ct}} = r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}}$ where $s'' = s' + s$ and $r'' = r' + r$. Besides, since $\widetilde{T}$ is equal to $T_I = s' \star X_I$, $\widetilde{\mathsf{ct}}$ is equal to $\mathsf{ct}_I = r' \star_{\mathsf{pk}} (-I \star_{\mathsf{M}} \mathsf{ct})$ and $\widetilde{\mathsf{C}} = \mathcal{O}(\mathsf{Com} \,\|\, \widetilde{T} \,\|\, \widetilde{\mathsf{ct}} \,\|\, \mathsf{bits})$ is equal to the leaf $\widetilde{\mathsf{C}} = \mathsf{C}_I \in \{\mathsf{C}_1, \cdots, \mathsf{C}_N\}$, the verifier reconstructs the root $\widetilde{\mathsf{root}}$ which is equal to root. Hence, the protocol has (non-abort) correctness.

<u>Abort Rate</u>. The prover will not abort in the case $\mathsf{chall} = 0$. When $\mathsf{chall} = 1$ (which occurs with probability $1/2$) the prover aborts if $s'' = s' + s \notin S_3$ or $r'' = r' + r \notin \overline{S}_3$. We note that s is in S_1 and s' is drawn uniformly at random from S_2 (in the random oracle model). We can therefore say s'' is drawn uniformly at random from $S_2 + s$, which contains S_3 as a subset. So the probability that $s'' = s' + s \in S_3$ is $|S_3| / |S_2| = \delta_x$. The same reasoning applies to r'', so the probability of both s'', r'' lying in $S_3, \overline{S}_3$ respectively is $\delta_x\delta_y$ and the total abort rate is $(1 - \delta_x\delta_y)/2$

<u>Relaxed Special Soundness</u>. Given two valid transcripts for the same statement and on the same commitment, $(\mathsf{com}, 0, \mathsf{seed})$ and $(\mathsf{com}, 1, (s'', r'', \mathsf{path}, \mathsf{bits}))$ where $\mathsf{com} = \mathsf{root}$, an extraction algorithm $\mathsf{Extract}$ for a witness for the relation $\tilde{R}'_{\mathsf{pk}}$ proceeds as follows. $\mathsf{Extract}$ firstly generates $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand} \,\|\, \mathsf{seed})$ and constructs $\mathsf{C}_1, \cdots, \mathsf{C}_N$ such that the Merkle Tree with leaves $(\mathsf{C}_1, \cdots, \mathsf{C}_N)$ has root equal to root. $\mathsf{Extract}$ outputs $\mathsf{W} = (\mathsf{Coll} \,\|\, x_1, \mathsf{Coll} \,\|\, x_2)$ as the witness if it there exists $x_1 \neq x_2$ such that $\mathcal{O}(\mathsf{Coll} \,\|\, x_1) = \mathcal{O}(\mathsf{Coll} \,\|\, x_2)$. Otherwise, by Lem. A.1, we can assume $\widetilde{\mathsf{C}} = \mathcal{O}(\mathsf{Com} \,\|\, s'' \star X_0 \,\|\, r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}} \,\|\, \mathsf{bits})$ is equal to $\mathsf{C}_{\widetilde{I}} \in \{\mathsf{C}_1, \cdots, \mathsf{C}_N\}$ for some $\widetilde{I} \in [N]$. Then, $\mathsf{Extract}$ outputs $\mathsf{W} = (\mathsf{Com} \,\|\, x_1, \mathsf{Com} \,\|\, x_2)$ as the witness if it there exists $x_1 \neq x_2$ such that $\mathcal{O}(\mathsf{Com} \,\|\, x_1) = \mathcal{O}(\mathsf{Com} \,\|\, x_2)$. Otherwise, from $\widetilde{\mathsf{C}} = \mathsf{C}_{\widetilde{I}}$, we can assume $s' \star X_{\widetilde{I}} = s'' \star X_0$, $r' \star_{\mathsf{pk}} (\widetilde{I} \star_{\mathsf{M}} \mathsf{ct}) = r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}}$, and $\mathsf{bits} = \mathsf{bits}_{\widetilde{I}}$. Let $\widetilde{s} = -s' + s'' \in S_2 + S_3$ and $\widetilde{r} = -r' + r'' \in \overline{S}_2 + \overline{S}_3$. Finally, $\mathsf{Extract}$ outputs $\mathsf{W} = (\widetilde{I}, \widetilde{s}, \widetilde{r})$. Here, the equalities $\widetilde{s} \star X_0 = X_{\widetilde{I}}$ and $\widetilde{I} \star_{\mathsf{M}} (\widetilde{r} \star_{\mathsf{pk}} Y_{\mathsf{pk}}) = \mathsf{ct}$ follow directly from the relations $s' \star X_{\widetilde{I}} = s'' \star X_0$ and $r' \star_{\mathsf{pk}} (\widetilde{I} \star_{\mathsf{M}} \mathsf{ct}) = r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}}$, respectively. Therefore, $\mathsf{W} = (\widetilde{I}, \widetilde{s}, \widetilde{r})$ is a witness for the "relaxed" relation $\tilde{R}'_{\mathsf{pk}}$. Hence, the protocol $\Pi_{\Sigma}^{\mathsf{base}}$ has relaxed special soundness. ☐

Theorem 5.2. *The sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$ has non-abort special zero-knowledge. Precisely, there exists a PPT simulator $\mathsf{Sim}^{\mathcal{O}}$ with access to a random oracle $\mathcal{O}$ such that, for any statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, $\mathsf{chall} \in \{0, 1\}$, *and any computationally-unbounded adversary $\mathcal{A}$ that makes at most Q queries to the random oracle $\mathcal{O}$, we have*

$$\left| \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) = 1] \right| \leq \frac{Q}{2^\lambda},$$

where $\widetilde{P}$ is a non-aborting *prover $P' = (P'_1, P'_2)$ run on* (X, W) *with a challenge fixed to* chall.

Proof. Assume the adversary makes Q_{Expand} and Q_{Com} queries to the random oracles of the form $\mathcal{O}(\mathsf{Expand} \,\|\, \cdot)$ and $\mathcal{O}(\mathsf{Com} \,\|\, \cdot)$, respectively. We have $Q_{\mathsf{Expand}} + Q_{\mathsf{Com}} \leq Q$. The PPT simulator $\mathsf{Sim}^{\mathcal{O}}$, on input $(\mathsf{X}, \mathsf{chall})$, proceeds as follows.

- If $\mathsf{chall} = 0$, the simulator executes as $P'^{\mathcal{O}}(\mathsf{X}, \perp, \mathsf{chall})$, where notice P' does not require the witness when $\mathsf{chall} = 0$. Concretely, the simulator outputs $(\mathsf{com} = \mathsf{root}, \mathsf{chall} = 0, \mathsf{resp} = \mathsf{seed})$ where $\mathsf{root}, \mathsf{seed}$ are honestly generated as in the execution of $P_1'^{\mathcal{O}}$.
- If $\mathsf{chall} = 1$, the simulator uniformly samples (s'', r'') from $S_3 \times \overline{S}_3$, and bits from $\{0,1\}^\lambda$. It computes $\mathsf{C}_1 = \mathcal{O}(\mathsf{Com} \parallel s'' \star X_0 \parallel r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}} \parallel \mathsf{bits})$. It then uniformly samples dummy commitments C_i for $i \in \{2, \ldots, N\}$ from $\{0,1\}^{2\lambda}$, and computes the (index-hiding) Merkle tree $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \ldots, \mathsf{C}_N)$. After that, it extracts the path $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, 1)$ in the tree and sets $\mathsf{com} = \mathsf{root}$, and $\mathsf{resp} = (s'', r'', \mathsf{path}, \mathsf{bits})$. Finally, the simulator returns $(\mathsf{com}, \mathsf{chall} = 1, \mathsf{resp})$.

In the first case, the whole transcript is generated exactly as in the protocol. Hence transcripts generated by $\widetilde{P}^{\mathcal{O}}$ and $\mathsf{Sim}^{\mathcal{O}}$ are indistinguishable to the adversary $\mathcal{A}$. Therefore, we have

$$\left| \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall} = 0)) = 1] \right| = \left| \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall} = 0)) = 1] \right| .$$

To conclude the proof, it suffices to show that the difference between the probabilities that the adversary $\mathcal{A}$ outputs 1 for the other case, $\mathsf{chall} = 1$, is also bounded by $\frac{Q}{2^\lambda}$.

We use a hybrid argument by introducing a series of simulators $\mathsf{Sim}_0 = \widetilde{P}, \ldots, \mathsf{Sim}_4 = \mathsf{Sim}$, gradually changing from the honest prover $\widetilde{P}$ to Sim, to show that they are indistinguishable with overwhelming probability. We fix an adversary $\mathcal{A}$, $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, and for each $i \in \{0, 1, \ldots, 4\}$, we denote by E_i the event that $\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}_i^{\mathcal{O}}(\mathsf{X}, \mathsf{chall} = 1)) = 1$.

- Sim_1 is identical to Sim_0 except that instead of using Expand to generate $s', r', \{\mathsf{bits}_i\}_{i \in [N]}$, the simulator generates these by sampling uniformly at random from the corresponding domains. This does not change the view of $\mathcal{A}$, unless the adversary queries $\mathcal{O}$ on input $(\mathsf{Expand} \parallel \mathsf{seed})$. Since seed has λ bits of min-entropy and because it is information-theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{A}$ queries $\mathcal{O}$ on this input is bounded by $Q_{\mathsf{Expand}}/2^\lambda$. That is, $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| \leq \frac{Q_{\mathsf{Expand}}}{2^\lambda}$.
- Sim_2 is identical to Sim_2 except that all the commitments C_i for $i \in [N] \setminus \{I\}$ are generated uniformly at random. This does not change the view of $\mathcal{A}$, unless the adversary queries $\mathcal{O}$ on input $(\mathsf{Com} \| T_i \| \mathsf{ct}_i \| \mathsf{bits}_i)$ for any $i \in [N] \setminus \{I\}$, where $T_i = s' \star X_i$ and $\mathsf{ct}_i = r' \star_{\mathsf{pk}} (-i \star_{\mathsf{M}} \mathsf{ct})$. Since for any $i \in [N] \setminus \{I\}$ the string bits_i has λ bits of min-entropy and because it is information-theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{A}$ queries $\mathcal{O}$ on input $(\mathsf{Com} \parallel T_i \parallel \mathsf{ct}_i \parallel \mathsf{bits}_i)$ is bounded by $Q_{\mathsf{Com}}/2^\lambda$. That is, $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \leq \frac{Q_{\mathsf{Com}}}{2^\lambda}$.
- Sim_3 is identical to Sim_3 except that instead of computing s'', r'' as $s' + s, r' + r$ (conditioned on them respectively lying in $S_3, \overline{S}_3$, due to non-aborting transcripts), the simulator generates these two values by sampling uniformly at random from $S_3, \overline{S}_3$, respectively . Both the distributions are uniform over S_3 and $\overline{S}_3$. Therefore, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = 0$.
- $\mathsf{Sim}_4 = \mathsf{Sim}$ is identical to Sim_4 except that the simulator uses $I = 1$ instead of the value I in the witness W. These two simulators are indistinguishable because the Merkle tree is index-hiding (by Lemma A.2). Precisely, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| = 0$.

Collecting the bounds, we obtain the bound in the statement. □

5.2 From Base to Main Traceable OR Sigma Protocol

In this section, compile $\Pi_\Sigma^{\mathsf{base}}$ to make the soundness error negligibly small. This is straightforward if we run the OR sigma protocol in parallel λ-times. However, we show how to do much better by incorporating the three optimizations developed in [BKP20] explained in the technical overview. Our main traceable OR sigma protocol, denote by $\Pi_\Sigma^{\mathsf{tOR}}$, is detailed in Fig. 4.

```
round 1: $P_1'^{\mathcal{O}}((\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s, r))$
$\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
$(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand} \parallel \mathsf{seed})$    ▷ Sample $(s', r') \in S_2 \times \overline{S}_2$ and $\mathsf{bits} \in \{0,1\}^\lambda$
for $i$ from 1 to $N$ do
    $(T_i, \mathsf{ct}_i) \leftarrow (s' \star X_i, r' \star_{\mathsf{pk}} (-i \star_{\mathsf{M}} \mathsf{ct}))$
    $\mathsf{C}_i \leftarrow \mathcal{O}(\mathsf{Com} \parallel T_i \parallel \mathsf{ct}_i \parallel \mathsf{bits}_i)$    ▷ Create commitments $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N)$
Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.
```

```
round 2: $V_1'(\mathsf{com})$
$c \xleftarrow{\$} \{0,1\}$
Verifier sends $\mathsf{chall} \leftarrow c$ to Prover.
```

```
round 3: $P_2'((I, s, r), \mathsf{chall})$
$c \leftarrow \mathsf{chall}$
if $c = 1$ then
    $(s'', r'') \leftarrow (s' + s, r' + r)$
    if $s'' \notin S_3$ or $r'' \notin \overline{S}_3$ then
        $P$ aborts the protocol.
    $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
    $\mathsf{resp} \leftarrow (s'', r'', \mathsf{path}, \mathsf{bits}_I)$
else
    $\mathsf{resp} \leftarrow \mathsf{seed}$
Prover sends $\mathsf{resp}$ to Verifier
```

```
Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
$(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{chall})$
if $c = 1$ then
    $(s'', r'', \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{resp}$
    if $s'' \notin S_3$ or $r'' \notin \overline{S}_3$ then
        $V$ outputs $\mathsf{reject}$.
    $(\widetilde{T}, \widetilde{\mathsf{ct}}) \leftarrow (s'' \star X_0, r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}})$
    $\widetilde{\mathsf{C}} \leftarrow \mathcal{O}(\mathsf{Com} \parallel \widetilde{T} \parallel \widetilde{\mathsf{ct}} \parallel \mathsf{bits})$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
    Verifier accepts only if $\widetilde{\mathsf{root}} = \mathsf{root}$.
else
    Repeat round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
    Output $\mathsf{accept}$ if the computation results in $\mathsf{root}$,
    and $\mathsf{reject}$ otherwise.
```

Figure 3: Construction of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the relation R_{sig}. Informally, $O(\mathsf{Expand}\|\cdot)$ and $O(\mathsf{Com}\|\cdot)$ are a PRG and a commitment scheme instantiated by the random oracle, respectively.

Unbalanced Challenge Space. Given the construction $\Pi_\Sigma^{\mathsf{base}}$, one can observe that the response produced by the prover by running P_2' when the challenge is 1 is larger than the response produced when the challenge is 0, which is a single seed of λ bits. Concretely, the response for the challenge $\mathsf{chall} = 1$ consists of a Merkle tree path, two elements in $S_3, \overline{S}_3$ respectively, and a λ bit string. We leverage this fact by preparing an unbalanced challenge space $C_{M,K}$, where each element in $C_{M,K}$ is a string containing K 1's and $M - K$ 0's. We chose $K \ll M$ to chose more 0's, while satisfying $\binom{M}{K} \geq 2^\lambda$ for negligible soundness error.

Seed Trees. The seed tree described in App. A.2 allows the prover to generate all seeds seed by using a single seed $\mathsf{seed}_{\mathsf{root}}$, and reveal parts of the tree according to the challenge. A smaller signature size follows directly from this approach.

Adding Salt. We prefix a salt and the session identifier, i.e. $(\mathsf{salt} \parallel i)$, to the random oracle when used within the i-th parallel execution of $\Pi_\Sigma^{\mathsf{base}}$. In particular, throughout such execution, $\mathcal{O}_i(\cdot) = \mathcal{O}(\mathsf{salt} \parallel i \parallel \cdot)$ is used. The salt is used as a prefix also within the construction of Merkle trees and seed trees. Adding salt benefits the protocol in having a tighter reduction and resisting multi-target attacks, such as those in [DN19]. The approach appears to make no difference in a sigma protocol but it's quite benefitial to a group (ring) signature scheme after we apply Fiat-Shamir transform. Roughly, in the anonymity game (Def. 2.17) each oracle $\mathcal{O}$ query made by the adversary will only give useful information to at most one challenge signature due to distinct prefix salts. In contrast, without salts an oracle query of $\mathcal{O}$ can give useful information to *each* challenge signature.

```
round 1: $P_1^{\mathcal{O}}((\{X_i\}_{i \in [N]}, \mathsf{pk}, \mathsf{ct}), (I, s, r))$
    $\mathsf{seed}_{\mathsf{root}} \leftarrow \{0,1\}^{\lambda}$
    $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$
    $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt} \parallel 0 \parallel \cdot)$
    $(\mathsf{seed}_1, \cdots, \mathsf{seed}_M) \leftarrow \mathsf{SeedTree}^{\mathcal{O}'}(\mathsf{seed}_{\mathsf{root}}, M)$
    for $j$ from 1 to $M$ do
        $\mathcal{O}_j(\cdot) := \mathcal{O}(\mathsf{salt} \parallel j \parallel \cdot)$
        $\mathsf{com}_j \leftarrow P_1'^{\mathcal{O}_j}((\{X_i\}_{i \in [N]}, \mathsf{pk}, \mathsf{ct}), (I, s, r); \mathsf{seed}_j)$
    Prover sends $\mathsf{com} \leftarrow (\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M)$ to Verifier.

round 2: $V_1(\mathsf{com})$
    $\mathbf{c} \xleftarrow{\$} C_{M,K}$
    Verifier sends $\mathsf{chall} \leftarrow \mathbf{c}$ to Prover.

round 3: $P_2^{\mathcal{O}}((I, s, r), \mathsf{chall})$
    $\mathbf{c} = (c_1, \cdots, c_M) \leftarrow \mathsf{chall}$
    $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt} \parallel \cdot)$
    for $j$ s.t. $c_j = 1$ do
        $\mathsf{resp}_j \leftarrow P_2'^{\mathcal{O}_j}((I, s, r), c_j; \mathsf{seed}_j)$        ▷ Run $P_2'$ on state $\mathsf{seed}_j$
    $\mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{ReleaseSeeds}^{\mathcal{O}'}(\mathsf{seed}_{\mathsf{root}}, 1^M \oplus \mathbf{c})$
    Prover sends $\mathsf{resp} \leftarrow (\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{resp}_j\}_{j \text{ s.t. } c_j = 1})$ to Verifier

Verification: $V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
    $((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M), \mathbf{c} = (c_1, \cdots, c_M)) \leftarrow (\mathsf{com}, \mathsf{chall})$
    $(\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{resp}_j\}_{j \text{ s.t. } c_j = 1}) \leftarrow \mathsf{resp}$
    $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt} \parallel 0 \parallel \cdot)$
    $\{\mathsf{resp}_j\}_{j \text{ s.t. } c_j = 0} \leftarrow \mathsf{RecoverLeaves}^{\mathcal{O}'}(\mathsf{seeds}_{\mathsf{internal}}, 1^M \oplus \mathbf{c})$
    for $j$ from 1 to $M$ do
        $\mathcal{O}_j(\cdot) := \mathcal{O}(\mathsf{salt} \parallel j \parallel \cdot)$
        Verifier outputs reject if $V_2'^{\mathcal{O}_j}(\mathsf{com}_j, c_j, \mathsf{resp}_j)$ outputs reject.
    Verifier outputs accept.
```

Figure 4: Main traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}} = (P = (P_1, P_2), V = (V_1, V_2))$ for the relation R_{sig} built on the the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ in Fig. 3. The challenge space is defined as $C_{M,K} := \{\mathbf{c} \in \{0,1\}^M \mid \|\mathbf{c}\|_1 = K\}$. Both the seed tree and $\Pi_{\Sigma}^{\mathsf{base}}$ have access to salted random oracles derived from $\mathcal{O}$.

Theorem 5.3. *The sigma protocol* $\Pi_\Sigma^{\mathsf{tOR}}$ *has correctness with abort rate* $(1-\delta_x^K\delta_y^K)$*, high min-entropy, and relaxed special soundness for the relations* R_{sig} *and* $\tilde{R}'_{\mathsf{sig}}$*, where the relations are identical to those used in Thm. 5.1.*

Proof. As a starting remark, we note that in the following lines we will use the notation of Fig. 4.

Correctness and Abort Rate. If the execution of $\Pi_\Sigma^{\mathsf{tOR}}$ does not abort, then the verifier will accept with probability 1 due to the correctness of $\Pi_\Sigma^{\mathsf{base}}$ and $\mathsf{SeedTree}$. We recall that in the case of challenge equal to 1 the execution of $\Pi_\Sigma^{\mathsf{base}}$ will abort with probability $(1-\delta_x\delta_y)$. Since the challenge $\mathbf{c}$, sampled from $C_{M,K}$, is of Hamming weight K, the abort rate of $\Pi_\Sigma^{\mathsf{tOR}}$ is $(1-\delta_x^K\delta_y^K)$.

High Min-Entropy. Since a random salt of length 2λ is included in the commitment com, it has at least 2λ bits of min-entropy.

Relaxed Special Soundness. The proof is similar to the one for the relaxed special soundness of $\Pi_\Sigma^{\mathsf{base}}$. Let $(\mathsf{com}, \mathsf{chall} = \mathbf{c}, \mathsf{resp})$ $(\mathsf{com}, \mathsf{chall}' = \mathbf{c}', \mathsf{resp}')$ be two accepting transcripts for the same statement. Without loss of generality, say $c_j = 0, c'_j = 1$, i.e. the j^{th} components of $\mathbf{c}$ and $\mathbf{c}'$ are different. By computing $\{\mathsf{resp}_i\}_{i \text{ s.t. } c_i=0} \leftarrow \mathsf{RecoverLeaves}^{\mathcal{O}'}(\mathsf{seeds}_{\mathsf{internal}}, 1^M \oplus \mathbf{c})$, the extraction algorithm gets resp_j. In this way, two valid transcripts $(\mathsf{com}_j, 0, \mathsf{resp}_j)$ and $(\mathsf{com}_j, 1, \mathsf{resp}'_j)$ for $\Pi_\Sigma^{\mathsf{base}}$ have been obtained, and the extractor of $\Pi_\Sigma^{\mathsf{base}}$ in Thm. 5.1 can be invoked to extract the witness for the relation $\tilde{R}_{\mathsf{sig}}$. To be concrete, in case a witness $\mathsf{W} = (x_1, x_2)$ is extracted by the extractor of Thm. 5.1 such that it forms a collision in the random oracle $\mathcal{O}_j = \mathcal{O}(\mathsf{salt} \,\|\, j \,\|\, \cdot)$, then the extractor appends x_1 and x_2 by either $\mathsf{salt} \,\|\, j \,\|\, \mathsf{Coll}$ or $\mathsf{salt} \,\|\, j \,\|\, \mathsf{Com}$ to produce a collision in $\mathcal{O}$.

□

Theorem 5.4. *The sigma protocol* $\Pi_\Sigma^{\mathsf{tOR}}$ *has non-abort special zero-knowledge. Precisely, there exists a PPT simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle* $\mathcal{O}$ *such that, for any statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$*,* $\mathsf{chall} \in C_{M,K}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most* Q *queries of the form* $(\mathsf{salt} \,\|\, \cdot)$ *to the random oracle* $\mathcal{O}$*, where* salt *is the salt value included in the transcript returned by* $\widetilde{P}$ *or* Sim*, we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) = 1]\right| \le \frac{Q}{2^\lambda},$$

where $\widetilde{P}$ *is a* non-aborting *prover* $P = (P_1, P_2)$ *run on* (X, W) *with a challenge fixed to* chall.

Proof. The PPT simulator $\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})$ for the main sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$ proceeds as in Fig. 5, where the simulator used for the base sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ in Thm. 5.2, denoted by Sim' is a subroutine. Say the adversary makes Q_i queries to the random oracle of the form $\mathcal{O}(\mathsf{salt}\|i\|\cdot)$ for $i \in \{0\} \cup [M]$. We have $\Sigma_0^M Q_i \le Q$.

We use a hybrid argument by introducing a sequence of simulators $\mathsf{Sim}_0, \cdots, \mathsf{Sim}_2$ that gradually change from $\mathsf{Sim}_0 = \widetilde{P}$ to $\mathsf{Sim}_2 = \mathsf{Sim}$. We fix an adversary $\mathcal{A}$, $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, and for each $i \in \{0, 1, 2\}$, we denote by E_i the event $\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}_i^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) = 1$.

- Sim_1 is identical to Sim_0, except that, rather than using a $\mathsf{SeedTree}$ with root $\mathsf{seed}_{\mathsf{root}}$ to generate $\mathsf{seeds}_{\mathsf{internal}}$ and $\{\mathsf{seed}_i\}_{i \text{ s.t. } c_i=0}$, the simulator instead runs $\mathsf{SimulateSeeds}(1^M \oplus \mathbf{c})$ to obtain $\mathsf{seeds}_{\mathsf{internal}}$, and then $\{\mathsf{seed}_i\}_{i \text{ s.t. } c_i=0}$ via $\mathsf{RecoverLeaves}(\mathsf{seeds}_{\mathsf{internal}}, 1^M \oplus \mathbf{c})$. The simulator picks the remaining seeds (for the challenge components c_i equal to 1) $\{\mathsf{seed}_i\}_{i \text{ s.t. } c_i=1}$ uniformly at random from $\{0,1\}^\lambda$. Lemma A.3 for the bit string $1^M \oplus \mathbf{c}$ implies that the distributions of $\mathsf{seeds}_{\mathsf{internal}}$ and $\{\mathsf{seed}_i\}_{i \text{ s.t. } c_i=1}$ generated in this way rather than as in the honest protocol can be distinguished with an advantage not greater than $\frac{Q_0}{2^\lambda}$. That is, $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| \le \frac{Q_0}{2^\lambda}$.

```
$\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})$
$\mathbf{c} = (c_1, \cdots, c_M) \leftarrow \mathsf{chall}$
$\mathsf{salt} \leftarrow \{0,1\}^{2\lambda}$
$\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt} \parallel 0 \parallel \cdot)$
$\mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{SimulateSeeds}^{\mathcal{O}'}(1^M \oplus \mathbf{c})$
$\{\mathsf{seed}_i\}_{i \text{ s.t. } c_i = 0} \leftarrow \mathsf{RecoverLeaves}^{\mathcal{O}'}(\mathsf{seeds}_{\mathsf{internal}}, 1^M \oplus \mathbf{c})$
for $i \in [M]$ do
    $\mathcal{O}'_i(\cdot) := \mathcal{O}(\mathsf{salt} \parallel i \parallel \cdot)$
    if $c_i = 1$ then
        $\mathsf{seed}_i \leftarrow \{0,1\}^{\lambda}$
    $(\mathsf{com}_i, c_i, \mathsf{resp}_i) \leftarrow \mathsf{Sim}'^{\mathcal{O}'_i}(\mathsf{X}, c_i; \mathsf{seed}_i)$
$\mathsf{com} \leftarrow (\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M)$
$\mathsf{resp} \leftarrow (\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{resp}_i\}_{i \text{ s.t. } c_i = 1})$
return $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
```

Figure 5: Zero-knowledge simulator Sim for the main sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}}$

- Sim_2 is identical to Sim_1 except that the simulator uses the base simulator subroutine Sim' to compute, for each $i \in [M]$ such that $c_i = 1$, com_i and resp_i on randomness bits_i by $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^{\lambda}$. By Theorem 5.2, the distinguishing advantage of the adversary is bounded by $\frac{Q_i}{2^{\lambda}}$ for each $i \in [M]$ such that $c_i = 1$. That is, $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| \leq \frac{\Sigma_1^M Q_i}{2^{\lambda}}$.

Collecting the bounds, we obtain the bound in the statement. □

5.3 Base Sigma Protocol for The "Tight" Relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$

In this section, we show how to slightly tweak our base sigma protocol for the relation R_{sig} to obtain a sigma protocol for the "tight" relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$ (see Sec. 3.3). This can then be used to construct the desired NIZK for $R_{\mathsf{sig}}^{\mathsf{Tight}}$ required for our tightly secure accountable ring signature construction.

As explained in the technical overview, we can use the sigma protocol for R_{sig} along with the sequential OR-proof [FHJ20] to construct a sigma protocol for the "tight" relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$. Unfortunately, this approach requires to double the proof size. Instead, we present a small tweak to our sigma protocol for R_{sig} to directly support statements in $R_{\mathsf{sig}}^{\mathsf{Tight}}$. Concretely, we use the same Merkle tree to commit to the $2N$ instances $\{X_i^{(j)}\}_{(i,j)\in[N]\times[2]}$ and for each $X_i^{(1)}$ and $X_i^{(2)}$, we encrypt the *same* index i. The main observation is that when the prover opens to the challenge bit 1 (which is the only case that depends on the witness), the path does no leak which $X_i^{(1)}$ and $X_i^{(2)}$ it opened to, and hence hides $b \in [2]$.

Notice the only increase in the size of the response is due to the path. Since the accumulated commitment only grows from N to $2N$, the overhead in the size of the path is merely 2λ bits. By using the unbalanced challenge space $C_{M,K}$ for the optimized parallel repetition, which consists of M-bit strings of Hamming weight K, the additional cost is only $2K\lambda$ where we typically set K to be a small constant (e.g., $K \leq 20$ for our concrete instantiation). This is much more efficient than the generic approach that doubles the proof size.

Formally, the sigma protocol for the "tight" relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$, denoted as $\Pi_{\Sigma}^{\mathsf{baseTi}}$, is provide in Fig. 6. We can turn it into a full-fledged sigma protocol with negligible soundness error by applying exactly the same argument in Sec. 5.1. We omit the proof of correctness and security for $\Pi_{\Sigma}^{\mathsf{baseTi}}$ as they are almost identical to those of our sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$ for R_{sig}.

```
round 1: $P_1'^{\mathcal{O}}((\{X_i^{(1)}, X_i^{(2)}\}_{i \in [N]}, \mathsf{pk}, \mathsf{ct}), (I, b, s, r))$
$\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
$(s', r', \boxed{\mathsf{bits}_1, \cdots, \mathsf{bits}_{2N}}) \leftarrow \mathcal{O}(\mathsf{Expand} \parallel \mathsf{seed})$    ▷ Sample $(s', r') \in S_2 \times \overline{S}_2$ and $\mathsf{bits} \in \{0,1\}^\lambda$
for $i$ from 1 to $N$ do
    $\mathsf{ct}_i \leftarrow r' \star_{\mathsf{pk}} (-i \star_{\mathsf{M}} \mathsf{ct})$
    for $b \in \{1, 2\}$ do
        $\boxed{T_{2(i-1)+b} \leftarrow s' \star X_i^{(b)}}$
        $\boxed{\mathsf{C}_{2(i-1)+b} \leftarrow \mathcal{O}(\mathsf{Com} \parallel T_{2(i-1)+b} \parallel \mathsf{ct}_i \parallel \mathsf{bits}_{2(i-1)+b})}$
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\boxed{\mathsf{C}_1, \cdots, \mathsf{C}_{2N}})$
Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.

round 2: $V_1'(\mathsf{com})$
$c \xleftarrow{\$} \{0,1\}$
Verifier sends $\mathsf{chall} \leftarrow c$ to Prover.

round 3: $P_2'((I, b, s, r), \mathsf{chall})$
$c \leftarrow \mathsf{chall}$
if $c = 1$ then
    $(s'', r'') \leftarrow (s' + s, r' + r)$
    if $s'' \notin S_3$ or $r'' \notin \overline{S}_3$ then
        $P$ aborts the protocol.
    $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, \boxed{2(I-1)+b})$
    $\mathsf{resp} \leftarrow (s'', r'', \mathsf{path}, \mathsf{bits}_I)$
else
    $\mathsf{resp} \leftarrow \mathsf{seed}$
Prover sends $\mathsf{resp}$ to Verifier

Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
$(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{chall})$
if $c = 1$ then
    $(s'', r'', \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{resp}$
    if $s'' \notin S_3$ or $r'' \notin \overline{S}_3$ then
        $V$ outputs $\mathsf{reject}$.
    $(\widetilde{T}, \widetilde{\mathsf{ct}}) \leftarrow (s'' \star X_0, r'' \star_{\mathsf{pk}} Y_{\mathsf{pk}})$
    $\widetilde{\mathsf{C}} \leftarrow \mathcal{O}(\mathsf{Com} \parallel \widetilde{T} \parallel \widetilde{\mathsf{ct}} \parallel \mathsf{bits})$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
    Verifier outputs $\mathsf{accept}$ if $\widetilde{\mathsf{root}} = \mathsf{root}$ and $\mathsf{reject}$ otherwise.
else
    Verifier repeats the computation of round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
    Verifier outputs $\mathsf{accept}$ if the computation results in $\mathsf{root}$, and $\mathsf{reject}$ otherwise.
```

Figure 6: Construction of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{baseTi}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the "tight" relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$. The box indicates the difference between the "non-tight" relation R_{sig}. Informally, $O(\mathsf{Expand} \| \cdot)$ and $O(\mathsf{Com} \| \cdot)$ are a PRG and a commitment scheme instantiated by the random oracle, respectively.

6 Multi-Proof Online Extractable NIZK From Sigma Protocol $\Pi_\Sigma^{\mathsf{tOR}}$

In this section, we show that applying the Fiat-Shamir transform to our traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$ from the previous section results in a multi-proof online extractable NIZK with labels $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$. The construction of our $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ for the relation R_{sig} is provide in Fig. 7.[9] We assume the output of $\mathcal{O}(\mathsf{FS} \parallel \cdot)$ is an M-bit string of Hamming weight K, i.e., the image is the challenge set $C_{M,K}$.

```
$\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, (\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s_I, r))$
  $\mathsf{resp} := \bot$
  while $\mathsf{resp} = \bot$ do
      $\mathsf{com} \leftarrow P_1^{\mathcal{O}}((\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s_I, r))$
      $\mathsf{chall} \leftarrow \mathcal{O}(\mathsf{FS} \parallel \mathsf{lbl} \parallel (\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \parallel \mathsf{com})$
      $\mathsf{resp} \leftarrow P_2^{\mathcal{O}}((I, s_I, r), \mathsf{chall})$
  return $\pi \leftarrow (\mathsf{com}, \mathsf{chall}, \mathsf{resp})$

$\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, (\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \pi)$
  $(\mathsf{com}, \mathsf{chall}, \mathsf{resp}) \leftarrow \pi$
  if $\mathsf{accept} \leftarrow V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{resp}) \wedge \mathsf{chall} = \mathcal{O}(\mathsf{FS} \parallel \mathsf{lbl} \parallel (\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \parallel \mathsf{com})$ then
      return $\top$
  else
      return $\bot$
```

Figure 7: A multi-proof online extractable NIZK with labels $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ for the relation R_{sig} obtained by applying the Fiat-Shamir transform to the traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}} = (P = (P_1, P_2), V = (V_1, V_2))$ in Fig. 4.

Correctness of $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ for the relation R_{sig} follows directly from the correctness of the underlying traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$. We show in Thms. 6.1 and 6.4 that $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ is multi-proof online extractable and zero-knowledge. We highlight that while we shown special soundness for $\Pi_\Sigma^{\mathsf{tOR}}$ with respect to the relaxed relation $\tilde{R}'_{\mathsf{sig}}$ (see Thm. 5.1), $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ is multi-proof online extractable with respect to the relaxed relation $\tilde{R}_{\mathsf{sig}}$ originally considered in Sec. 3.1 for the generic construction of accountable ring signature. At a high level, we upper bound the probability that a cheating prover finds a collision in the random oracle, which was the only difference between $\tilde{R}_{\mathsf{sig}}$ and $\tilde{R}'_{\mathsf{sig}}$. This subtle difference makes the resulting NIZK more handy to use as a building block, since we can ignore the edge case where the extractor accidentally extracts a collision in the random oracle. Below, we provide the proof of the multi-proof online extractability.

Theorem 6.1. *The* NIZK *with labels* $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ *in Fig. 7 is multi-proof online extractable for the family of relations* R_{sig} *and* $\tilde{R}_{\mathsf{sig}}$ *considered in Sec. 3.1, where* R_{sig} *was formally redefined using notations related to group actions in Sec. 5.1 and* $\tilde{R}_{\mathsf{sig}}$ *is formally redefined as follows:*

$$\tilde{R}_{\mathsf{sig}} = \left\{ ((\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W}) \;\middle|\; \begin{array}{c} \mathsf{W} = (I, s, r) \in [N] \times (S_2 + S_3) \times (\overline{S}_2 + \overline{S}_3) \\ \wedge\ X_I = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}.$$

More precisely, for any (possibly computationally-unbounded) adversary $\mathcal{A}$ *making at most* Q *queries to the random oracle and* T *queries to the extract oracle, we have*

$$\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK},\mathsf{lbl}}}(\mathcal{A}) \leq T \cdot \left(Q^2 / 2^{2\lambda - 2} + (M \cdot Q)/2^{\lambda} + 1/\left| C_{M,K} \right| \right),$$

where $C_{M,K}$ *is the challenge space (or equivalently the output space of* $\mathcal{O}(\mathsf{FS} \parallel \cdot)$*).*

---

[9] An astute reader may notice that the prover is only *expected* polynomial time. We can always assign an upper bound on the runtime of the prover, but did not do so for better readability. In practice, for concrete choices of the parameter, the number of repetition never exceeds, say 10.

Proof. We begin the proof by providing the description of the online extractor $\mathsf{OnlineExtract}$. Below, it is given as input $(\mathsf{lbl}, \mathsf{X}, \pi, L_{\mathcal{O}})$, where π is guaranteed to be valid by definition.

1. It parses $(\{X_i\}_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \leftarrow \mathsf{X}$, $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}}) \leftarrow \pi$, $((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M), \mathbf{c} = (c_1, \cdots, c_M)) \leftarrow (\overline{\mathsf{com}}, \overline{\mathsf{chall}})$, $(\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{resp}_j\}_{j \text{ s.t. } c_j=1}) \leftarrow \overline{\mathsf{resp}}$, and $\mathsf{root}_j \leftarrow \mathsf{com}_j$ for $j \in [M]$.[10]
2. For $j \in [M]$ such that $c_j = 1$, it proceeds as follows:
   (a) It parses $(s''_j, r''_j, \mathsf{path}_j) \leftarrow \mathsf{resp}_j$.
   (b) For every $\big((\mathsf{salt} \,\|\, j \,\|\, \mathsf{Expand} \,\|\, \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\big) \in L_{\mathcal{O}}$, where $\mathsf{salt} \,\|\, j \,\|\, \mathsf{Expand}$ is fixed, it proceeds as follows:
      i. It sets $(s, r) = (s''_j - s', r''_j - r')$ and checks if $(s, r) \in (S_2 + S_3) \times (\overline{S}_2 + \overline{S}_3)$.
      ii. It then checks if there exists $I \in [N]$ such that $X_I = s \star X_0$ and $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r)$.
      iii. If all the check above passes, it returns $\mathsf{W} = (I, s, r)$.
3. If it finds no witness W of the above form, then it returns $\mathsf{W} = \perp$.

We analyze the probability of $\mathcal{A}$ winning the multi-proof online extractability game with the above online extractor $\mathsf{OnlineExtract}$. Below, P' and V' are the prover and verifier of the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$ in Fig. 3.

- We say a tuple $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}, \mathsf{chall}, \mathsf{resp})$ is valid if the following properties hold:
  - $\mathsf{chall} = 1$;
  - $V_2'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ outputs accept (i.e., it is a valid transcript for $\Pi_{\Sigma}^{\mathsf{base}}$ with challenge 1);
  - there exists $(\mathsf{seed}, s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)$ such that $\big((\mathsf{salt}\|j\|\mathsf{Expand}\|\mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\big) \in L_{\mathcal{O}}$, and if we execute $P_1'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}$ with randomness seed, it produces com. Here, we use the fact that $P_1'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}$ can be executed without the witness. By correctness of $\Pi_{\Sigma}^{\mathsf{base}}$, this implies that $(\mathsf{com}, 0, \mathsf{seed})$ is a valid transcript.
- We say a tuple $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}, \mathsf{chall}, \mathsf{resp})$ is $\mathsf{invalid}$ if $\mathsf{chall} = 1$, $V_2'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ outputs accept, but it is not valid.

Observe that if $\mathsf{input}_{\mathsf{base}}$ is valid, then the online extractor can recover a valid transcript $(\mathsf{com}, 0, \mathsf{seed})$ from $\mathsf{input}_{\mathsf{base}}$. Then, it can (informally) extract a witness by combining it with $(\mathsf{com}, 1, \mathsf{resp})$ and using the extractor from $\Pi_{\Sigma}^{\mathsf{base}}$ constructed in Thm. 5.1. In contrast, if $\mathsf{input}_{\mathsf{base}}$ is $\mathsf{invalid}$, then intuitively, no adversary would be able to prepare a valid response $\mathsf{resp} = \mathsf{seed}$ for the challenge $\mathsf{chall} = 0$ since $L_{\mathcal{O}}$ (i.e., the random oracle query the adversary makes) does not contain a valid response. However, to make this claim formal, we need to also take into account the fact that the adversary may learn non-trivial information about $\mathsf{resp} = \mathsf{seed}$ via the proof output by the prove query. That is, when the challenger runs $P^{\mathcal{O}}$, the adversary may learn non-trivial input/output pairs without directly querying the random oracle itself. In this case, even though no useful information is stored in $L_{\mathcal{O}}$, the adversary may still be able to forge a proof.

We formally show in Lem. 6.2 below that if an adversary $\mathcal{A}$ submits an extract query on a valid input $(\mathsf{lbl}, \mathsf{X}, \pi)$, then a valid $\mathsf{input}_{\mathsf{base}}$ must be included in π (i.e., it cannot consist of $\mathsf{input}_{\mathsf{base}}$ that are all $\mathsf{invalid}$). This allows us to argue that the online extractor will be able to recover two valid transcripts with overwhelming probability, which then further allows the online extractor to extract the witness by running the extractor for the special soundness of the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{base}}$.

---

[10] Throughout the proof, we use overlines for $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}})$ to indicate that it is a transcript of of $\Pi_{\Sigma}^{\mathsf{tOR}}$. We use resp_i without overlines to indicate elements of $\overline{\mathsf{resp}}$.

Lemma 6.2. *Assume an adversary $\mathcal{A}$ submits a total of T extract queries of the form $\{(\mathsf{lbl}_k, \mathsf{X}_k, \pi_k)\}_{k\in[T]}$, where every π_k is a valid proof including the same* salt *and satisfies $(\mathsf{lbl}_k, \mathsf{X}_k, \pi_k) \notin L_P$. Let $\{(\mathsf{com}_{k,j}, \mathsf{chall}_{k,j}, \mathsf{resp}_{k,j})\}_{j\in[M]}$ be the transcript of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ that the verification algorithm reconstructs when verifying π_k (see Line 7 of **Verification** $V_2^{\mathcal{O}}$ in Fig. 4). Then, with probability at least $1 - T \cdot \left(Q_{\mathsf{salt}}/2^{2\lambda-1} + (M \cdot Q_{\mathsf{salt}})/2^{\lambda} + 1/\left|C_{M,K}\right|\right)$, for all $k \in T$ there exists at least one $j \in [M]$ such that $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}_k, \mathsf{salt}, j, \mathsf{com}_{k,j}, \mathsf{chall}_{k,j} = 1, \mathsf{resp}_{k,j})$ is* valid*.*

Proof. For any $k \in [T]$, let us redefine $\pi_k = (\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}})$, $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}) = ((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M), \mathbf{c} = (c_1, \cdots, c_M))$ where $\mathbf{c} = \mathcal{O}(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \overline{\mathsf{com}})$, $\overline{\mathsf{resp}} = (\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{resp}_j\}_{j \text{ s.t. } c_j=1})$. Namely, we omit the subscript k for better readability. We consider two cases: (1) there exists $(\mathsf{lbl}, \mathsf{X}, \pi') \in L_P$ such that $\pi' = (\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}}')$ and $\overline{\mathsf{resp}}' \neq \overline{\mathsf{resp}}$ and (2) no such entry in L_P exists.

We consider the first case (1). This corresponds to the case where $\mathcal{A}$ reuses the proof π' obtained through the prove query by simply modifying the response. We claim that this cannot happen with overwhelming probability. Let $\overline{\mathsf{resp}}' = (\mathsf{seed}'_{\mathsf{internal}}, \{\mathsf{resp}'_j\}_{j \text{ s.t. } c_j=1})$. It is clear if $\mathsf{seed}'_{\mathsf{internal}}$ is different from $\mathsf{seeds}_{\mathsf{internal}}$, then $\mathcal{A}$ finds a collision in the random oracle. Since we use a seed tree to generate the randomness used in each base sigma protocol, we can very loosely upper bound the probability of $\mathcal{A}$ outputting such transcript for any $k \in [T]$ by $Q_{\mathsf{salt}}/2^{2\lambda}$. Similarly, consider $\mathsf{resp}'_j \neq \mathsf{resp}_j$ for some j such that $c_j = 1$. Then, it either finds a collision in $\mathcal{O}(\mathsf{Coll} \parallel \cdot)$ (used by the Merkle tree) or $\mathcal{O}(\mathsf{Com} \parallel \cdot)$. We can again very loosely upper bound the probability of $\mathcal{A}$ outputting such transcript for any $k \in [T]$ by $Q_{\mathsf{salt}}/2^{2\lambda}$. Thus, case (1) occurs with probability at most $Q_{\mathsf{salt}}/2^{2\lambda-1}$.

We next consider the second case (2). If $\overline{\mathsf{com}}$ included in π is the same as π', then $\overline{\mathsf{chall}}$ is the same challenge included in π since the challenge is generated as $\mathcal{O}(\mathsf{FS}\|\mathsf{lbl}\|\mathsf{X} \parallel \overline{\mathsf{com}})$. However, this results in a tuple that falls in the first case (1). Therefore, there exists no π' in L_P that contains the same $\overline{\mathsf{com}}$ as π. This, in particular, implies that the output $\overline{\mathsf{chall}} \leftarrow \mathcal{O}(\mathsf{FS}\|\mathsf{lbl}\|\mathsf{X} \parallel \overline{\mathsf{com}})$ is distributed uniform random from the view of $\mathcal{A}$ before it makes the hash query.

Now, for the sake of contradiction, we assume $\mathsf{input}_{\mathsf{base},j} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}_j, c_j, \mathsf{resp}_j)$ is invalid for all $j \in [M]$ such that $c_j = 1$. Let $L_{\mathcal{O}_P}$ be a list that contains all the inputs/outputs of the random oracle queries $\mathsf{Prove}^{\mathcal{O}}$ makes when the challenger answers the prove query made by $\mathcal{A}$. We prove the following corollary.

Corollary 6.3. *For any $j^* \in [M]$, if $\mathsf{input}_{\mathsf{base},j^*}$ is* invalid*, then either of the following holds:*

- *there exists no tuple $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N, \mathsf{seed})$ and $j' \in [M]$ such that $\big((\mathsf{salt} \| j' \| \mathsf{Expand} \| \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\big) \in L_{\mathcal{O}_P}$, but if we execute $P_1'^{\mathcal{O}(\mathsf{salt}\|j'\|\cdot)}$ with randomness* seed*, it produces com_{j^*};*
- *there exists such a tuple but* seed *retains λ-bits of min-entropy from the view of $\mathcal{A}$ except with probability at most $(MQ_{\mathsf{salt}})/2^{\lambda}$.*

Proof. Assume such an entry is found in $L_{\mathcal{O}_P}$. This corresponds to the case $\mathcal{A}$ is reusing com_{j^*} that was included in a proof π obtained through the prove query. Let $\{(\mathsf{com}'_j, c'_j, \mathsf{resp}'_j)\}_{j\in[M]}$ be the transcript of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ that the verification algorithm reconstructs from such π (see Line 7 of **Verification** $V_2^{\mathcal{O}}$ in Fig. 4), where $\mathsf{com}'_{j'} = \mathsf{com}_{j^*}$. Our current goal is to prove that $c'_{j'} = 1$ (i.e., seed was not used as a response). Since $\mathsf{com}'_{j'}$ and com_{j^*} are roots of a Merkle tree and the indices j' and j^* are used as prefix to the hash when constructing the roots, respectively, the probability of $\mathcal{A}$ outputting com_{j^*} such that $j' \neq j^*$ is upper bounded by $((M-1)Q_{\mathsf{salt}})/2^{\lambda}$. Below, we assume $j' = j^*$. Recall by definition of the online extractability game (see Def. 2.10), $\mathcal{A}$ runs the verification algorithm to check if π is valid. Therefore, if $\mathsf{input}_{\mathsf{base},j^*}$ is invalid, then we have $c'_{j'} = 1$. Otherwise, there must exist an entry $\big((\mathsf{salt} \parallel j^* \parallel \mathsf{Expand} \parallel \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\big) \in L_{\mathcal{O}}$, which contradicts that $\mathsf{input}_{\mathsf{base},j^*}$ is invalid. This further implies that $\mathsf{resp}'_{j'}$ does not include seed. Then, by Lem. A.3 regarding the seed tree, seed that was used to construct $\mathsf{com}_{j'} = \mathsf{com}_{j^*}$ is statistically hidden to the adversary with all but probability $Q_{\mathsf{salt}}/2^{\lambda}$. The proof is completed by collecting all the bounds. □

By Corollary 6.3, if $\mathsf{input}_{\mathsf{base},j}$ is $\mathsf{invalid}$, then $\mathcal{A}$ cannot prepare a valid response for the challenge $c_j = 0$ with all but probability at most $(MQ_{\mathsf{salt}})/2^{\lambda}$. This is because such response is either not recorded in both $L_{\mathcal{O}}$ and $L_{\mathcal{O}_P}$, or it is recorded in $L_{\mathcal{O}_P}$ but the seed retains λ-bits of min-entropy from the view of $\mathcal{A}$ except with probability $(MQ_{\mathsf{salt}})/2^{\lambda}$. Moreover, since $\overline{\mathsf{chall}}$ is statistically hidden to $\mathcal{A}$ before it queries the random oracle, the probability that $\overline{\mathsf{chall}}$ coincides with challenges for which $\mathcal{A}$ can open to is at most $1 - 1/\left|C_{M,K}\right|$, where recall $C_{M,K}$ is the challenge space (or equivalently the output space of $\mathcal{O}(\mathsf{FS} \parallel \cdot)$).

Taking the union bound and collecting all the bounds together, at least one of the $\mathsf{input}_{\mathsf{base}}$ must be valid with the probability stated in the statement. This completes the proof of the lemma. ☐

We are now prepared to analyze the probability that $\mathcal{A}$ wins the multi-proof online extractability game with the aforementioned online extractor $\mathsf{OnlineExtract}$. By Lem. 6.2, if $\mathcal{A}$ makes at most T extract queries, then by a simple union bound and using the inequality $\sum_i Q_{\mathsf{salt}_i} \leq Q$, with probability at least $1 - T \cdot \left((2Q)/2^{2\lambda} + (M \cdot Q)/2^{\lambda} + 1/\left|C_{M,K}\right|\right)$, all the $\mathsf{input}_{\mathsf{base}}$ included in the queried proof are valid. Then, by the definition of valid and the description of $\mathsf{OnlineExtract}$, $\mathsf{OnlineExtract}$ is able to extract two valid transcripts for all T proofs queried by $\mathcal{A}$. Recalling Thms. 5.1 and 5.3, $\mathsf{OnlineExtract}$ either succeeds in extracting a witness $\mathsf{W} = (I, s, r) \in [N] \times (S_2 + S_3) \times (\overline{S}_2 + \overline{S}_3)$ or a witness that consists of a collision in $\mathcal{O}(\mathsf{salt} \parallel j \parallel \mathsf{Coll} \parallel \cdot)$ or $\mathcal{O}(\mathsf{salt} \parallel j \parallel \mathsf{Com} \parallel \cdot)$ for some $j \in [M]$. Hence, with all but probability $Q^2/2^{2\lambda}$, $\mathsf{OnlineExtract}$ succeeds in extracting a witness $\mathsf{W} = (I, s, r)$ as desired, conditioned on all the $\mathsf{input}_{\mathsf{base}}$ included in the queried proof are valid. Collecting the bounds, we arrive at our statement. ☐

Theorem 6.4. *The* NIZK *with labels* $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$ *in Fig. 7 is zero-knowledge. Precisely, there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that, for any statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most* Q_1 *queries to* $\mathcal{O}$ *or* Sim_0*, and* Q_2 *queries to* Prove *or* $\mathcal{S}$*, we have*

$$\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK},\mathsf{lbl}}}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\mathcal{O},\mathsf{Prove}}(1^{\lambda}) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}_0,\mathcal{S}}(1^{\lambda}) = 1\right]\right| \leq \frac{Q_2 \cdot (Q_1 + Q_2)}{2^{2\lambda}} + \frac{Q_1}{2^{\lambda}}.$$

Proof. To prove the zero-knowledge property of $\Pi_{\mathsf{NIZK},\mathsf{lbl}} = (\mathsf{Prove}^{\mathcal{O}}, \mathsf{Verify}^{\mathcal{O}})$, we define a zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ in Fig. 8, where Sim_0 and Sim_1 share states, including a list L which is initially empty. At a high level, Sim_0 simulates the random oracle $\mathcal{O}$ in an on-the-fly manner but replaces certain queries for consistency with Sim_1. On the other hand, Sim_1 simulates the prover oracle using the simulator from the underlying sigma protocol, which we denote here by Sim_{Σ} (see Thm. 5.4), as a subroutine. Specifically, Sim_1 is given a valid statement $\mathsf{X} = (\{X_i\}_{i \in [N]}, \mathsf{pk}, \mathsf{ct})$, and samples a random challenge chall from the challenge space $C_{M,K}$, which is also the output space of $\mathcal{O}(\mathsf{FS} \parallel \cdot)$. It then runs Sim_{Σ} on challenge chall by providing it oracle access to Sim_0, and updates the list L accordingly. In Fig. 8, we denote by D_x the distribution of $\mathcal{O}(x)$, where the probability is taken over the random choice of the random oracle $\mathcal{O}$. Without loss of generality, we assume D_x to be efficiently sampleable.

```
$\mathsf{Sim}_0(x)$
if $x \in L$ then
    return $L[x]$
$y \leftarrow D_x$
$L[x] := y$
return $y$
```

```
$\mathsf{Sim}_1(\mathsf{lbl}, \mathsf{X})$
$\mathsf{chall} \leftarrow C_{M,K}$
$(\mathsf{com}, \mathsf{chall}, \mathsf{resp}) \leftarrow \mathsf{Sim}_{\Sigma}^{\mathsf{Sim}_0}(\mathsf{X}, \mathsf{chall})$
if $(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com}) \in L$ then
    return $\bot$
$L[(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com})] := \mathsf{chall}$
return $\pi \leftarrow (\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
```

Figure 8: Zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ for $\Pi_{\mathsf{NIZK},\mathsf{lbl}}$

To show the indistinguishability of $(\mathcal{O}, \mathsf{Prove})$ and $(\mathsf{Sim}_0, \mathcal{S})$, we use a hybrid argument by introducing an intermediate pair of simulators $(\mathsf{Sim}_0, \mathsf{Sim}_{\mathsf{int}})$, where $\mathsf{Sim}_{\mathsf{int}}$ is defined in Fig. 9. Let $\mathcal{S}_{\mathsf{int}}$, analog to Prove and $\mathcal{S}$, be an oracle that on input $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ returns $\bot$ if $\mathsf{lbl} \notin \mathsf{L} \vee (\mathsf{X}, \mathsf{W}) \notin R_{\mathsf{sig}}$ and otherwise returns $\mathsf{Sim}_{\mathsf{int}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$.

```
$\mathsf{Sim}_{\mathsf{int}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$
1: $\mathsf{com} \leftarrow P_1^{\mathsf{Sim}_0}(\mathsf{X}, \mathsf{W})$
2: $\mathsf{chall} \leftarrow C_{M,K}$
3: if $(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com}) \in L$ then
4:     return $\perp$
5: $L[(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com})] := \mathsf{chall}$
6: $\mathsf{resp} \leftarrow P_2^{\mathsf{Sim}_0}(\mathsf{X}, \mathsf{chall})$
7: return $\pi \leftarrow (\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
```

Figure 9: Intermediate simulator $\mathsf{Sim}_{\mathsf{int}}$, where $P = (P_1, P_2)$ is the prover of the traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}}$ in Fig. 4.

Suppose $\mathcal{A}$ makes Q_1 queries to the oracles $\mathcal{O}$ or Sim_0, and Q_2 queries to the oracles $\mathsf{Prove}, \mathcal{S}_{\mathsf{int}}$, or $\mathcal{S}$. For each $i \in \{1, 2, 3\}$, we denote by E_i the event that $\mathcal{A}$ returns 1 respectively. We analyze the differences by defining three games as follows:

Game_1 : This is the real zero-knowledge game where $\mathcal{A}$ is given access to $\mathcal{O}$ and Prove.

Game_2 : The game is modified to provide $\mathcal{A}$ access to Sim_0 and $\mathcal{S}_{\mathsf{int}}$ instead. The view of $\mathcal{A}$ is identical to the previous game unless $\mathsf{Sim}_{\mathsf{int}}$ outputs $\perp$ in Line 4. Roughly, this occurs when the reprogramming of the random oracle fails due to the input being already defined. By Thm. 5.3, com has 2λ bits of min-entropy. Since at most $Q_1 + Q_2$ queries of the form $(\mathsf{FS} \| \mathsf{lbl} \| \mathsf{X} \| \mathsf{com})$ are made in this game, we have $|\Pr[E_1] - \Pr[E_2]| \leq \frac{Q_2 \cdot (Q_1 + Q_2)}{2^{2\lambda}}$.

Game_3 : The game is modified to provide $\mathcal{A}$ access to Sim_0 and $\mathcal{S}$ instead. The only difference is that rather than computing honestly via (P_1, P_2) from the traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}}$, the simulator Sim_1 simulates these using the simulator Sim_{Σ} provided by $\Pi_{\Sigma}^{\mathsf{tOR}}$.

Let salt_i represent the salt that $\mathsf{Sim}_{\mathsf{int}}$ or Sim_1 samples on its i-th invocation. For $i \in [Q_2]$, let Q_i' be the number of queries the adversary makes to oracle Sim_0 of the form $(\mathsf{salt}_i \| \cdot)$. By Thm. 5.4, the advantage of the adversary in distinguishing $\mathsf{Sim}_{\mathsf{int}}$ or Sim_1 is bounded by $\frac{Q_i'}{2^\lambda}$ for each $i \in [Q_2]$. Therefore, $|\Pr[E_2] - \Pr[E_3]| \leq \frac{\sum_1^{Q_2} Q_i'}{2^\lambda} \leq \frac{Q_1}{2^\lambda}$

Collecting the bounds, we obtain the bound in the statement. ☐

7 Instantiations

We instantiate the building blocks required for our generic construction of an accountable ring signature scheme presented in Sec. 3 via isogenies based on CSIDH group action and lattices.

7.1 Instantiation from Isogenies

We instantiate a group-action-based HIG and PKE, and the corresponding NIZKs for the relations R_{sig} and R_{open} based on the CSIDH paradigm. In particular we assume that the structure of the ideal class group $\mathcal{C}\ell(\mathcal{O})$ is known, and cyclic of odd order n, so that it is isomorphic to $\mathbb{Z}_n$. Given a generator $\mathfrak{g}$ of $\mathcal{C}\ell(\mathcal{O})$, $\mathbb{Z}_n$ acts freely and transitively on $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ via the group action $\star : (a, E) \mapsto \mathfrak{g}^a * E$, which we can compute efficiently. Note that in case the class group structure is not known (e.g., at higher security levels where computing the class group is currently not feasible.) we can still instantiate all the building blocks using rejection sampling *à la* SeaSign.

Group-Action-Based HIG**.** We instantiate the group-action-based HIG defined by the algorithms $(\mathsf{RelSetup}, \mathsf{IGen})$ as follows. The output of $\mathsf{RelSetup}$ describes a setup for a CSIDH group action $\star : \mathcal{C}\ell(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}, \pi) \to \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, sets $G = S_1 = S_2 = \mathcal{C}\ell(\mathcal{O})$, $\delta = 1$, $\mathcal{X} = \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, and $X_0 = E_0$, where E_0 is the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$. The output of IGen is then $(E_0, \mathfrak{a} \star E_0)$, where $\mathfrak{a}$ is uniformly sampled from $\mathcal{C}\ell(\mathcal{O})$. Then the properties of Def. 4.1 are easily verified. In particular, the security of the hard instance generator is equivalent to the hardness of GAIP for CSIDH. Moreover, it is not difficult to see that the group-action-based HIG is also a hard *multi-instance* generator based on the same assumption. Concretely, given one instance (E_0, E), the reduction can rerandomize this arbitrarily many times to obtain fresh statements $(E_0, \mathfrak{b} \star E)$, where $\mathfrak{b}$ is uniformly sampled from $\mathcal{C}\ell(\mathcal{O})$. If an adversary succeeds in breaking any of these instances, then the reduction can subtract $\mathfrak{b}$ from it to solve its original instance.

Group-Action-Based PKE**.** We can define an ElGamal-like public-key encryption scheme $\Pi_{\mathsf{GA\text{-}PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ based on the CSIDH group action, as follows. Note that the decryption algorithm works by enumerating the message space, so the PKE is only efficient when the message space $\mathcal{M}$ (which is a subset of $\mathcal{C}\ell(\mathcal{O})$) is polynomially large. This relaxed notion of decryption suffices for our ARS generic construction.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input a security parameter 1^λ, it returns the setup for a CSIDH group action $\star : \mathcal{C}\ell(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}, \pi) \to \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, and sets $G = G_M = S_1 = S_2 = \mathcal{C}\ell(\mathcal{O}), \mathcal{X} = \mathcal{E}\ell\ell_p(\mathcal{O}, \pi) \times \mathcal{E}\ell\ell_p(\mathcal{O}, \pi), \delta_y = 1$. The "message" group action $\star_{\mathsf{M}} : G \times \mathcal{X} \to \mathcal{X}$ is defined as $(a, (E_1, E_2)) \mapsto (E_1, a \star E_2)$ (i.e., $\star_{\mathsf{M}}$ acts on the second component only).

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: On input a public parameter pp, it returns a secret key sk sampled uniformly from $\mathcal{C}\ell(\mathcal{O})$, and a public key $\mathsf{pk} = (\star_{\mathsf{pk}}, X_{\mathsf{pk}})$, where $\star_{\mathsf{pk}} : G \times \mathcal{X} \to \mathcal{X}$ is defined as $(a, (E_1, E_2)) \mapsto (a \star E_1, a \star E_2)$ (i.e., $\star_{\mathsf{pk}}$ acts on both components), and $X_{\mathsf{pk}} = \mathsf{sk} \star E_0$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{M}; r) \to \mathsf{ct}$: On input a public key $\mathsf{pk} = (\star_{\mathsf{pk}}, X_{\mathsf{pk}})$ and a message $\mathsf{M} \in \mathcal{M}$, it returns the ciphertext $\mathsf{ct} = (\mathsf{M} \star_{\mathsf{M}} (r \star_{\mathsf{pk}} Y_{\mathsf{pk}}) \in \mathcal{Y}$, where $r \leftarrow G$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$: On input a secret key sk and a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, the decryption algorithm tries all messages $\mathsf{M} \in \mathcal{M}$ until it finds a message M such that $\mathsf{M} \star \mathsf{ct}_1 = -\mathsf{sk} \star \mathsf{ct}_2$. If such a message exists, it is unique, and the algorithm outputs it; otherwise, $\perp$ is output.

It is not difficult to verify that the above-defined $\Pi_{\mathsf{GA\text{-}PKE}}$ is correct (with probability 1). The decryption scheme of $\Pi_{\mathsf{GA\text{-}PKE}}$ differs from that of ElGamal since it is not possible to *divide out* $sk \star \mathsf{ct}_1$ from ct_2. Therefore, retrieving M from $\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{sk}$ requires the resolution of an instance of GAIP with input $(\mathsf{sk} \star \mathsf{ct}_1, \mathsf{ct}_2)$. Dec solves this problem by a brute force over the message space $\mathcal{M}$. In case $\mathcal{M}$ is polynomially large, then we have efficient decryption as desired.

Multi-Challenge $\mathsf{IND\text{-}CPA}$ *Security.* The scheme is multi-challenge $\mathsf{IND\text{-}CPA}$ secure based on the dCSIDH assumption. Since $\Pi_{\mathsf{GA\text{-}PKE}}$ is an ElGamal-like encryption scheme in the CSIDH setting — where each exponentiation is replaced by a group action — for the security proof it is sufficient to adapt the usual proof for the group-based ElGamal encryption scheme. Note that the the reduction loses a factor $1/Q_{\mathsf{ct}}$, where Q_{ct} is the number of challenge ciphertext the adversary observes. This is the only reason why we do not achieve tight security for our accountable ring signature and group signature.

We point out that by ignoring the PKE, we obtain a ring signature identical to Beullens et al. [BKP20]. Thus we obtain the first tightly secure and efficient isogeny-based ring signature in this work.

$(\mathcal{R}', \mathcal{KR}')$*-correctness.* In the isogeny setting, it is not needed to relax the key relation (contrary to our lattice instantiation where some relaxation is necessary in order to get an efficient opening proof). We can simply set $\mathcal{KR}' = \mathcal{KR} = \{(E, \mathsf{sk}) \mid \mathsf{sk} \star E_0 = E\} \subseteq \mathcal{E}\ell\ell_p(\mathcal{O}, \pi) \times \mathcal{C}\ell$. Similarly, since $S_2 = S_1$, there is no relaxation in the encryption randomness. Therefore $(\mathcal{R}', \mathcal{KR}')$-correctness is equivalent to the standard correctness property (with probability 1), which is satisfied by our PKE.

Multi-Proof Online Extractable NIZK **with Labels** $\Pi_{\mathsf{NIZK,lbl}}$**.** Using the group-action-based HIG and PKE, we can instantiate $\Pi_{\mathsf{NIZK,lbl}}$ for the signing relation R_{sig} (see Sec. 3.1) as explained in Secs. 5 and 6.

Statistically Sound NIZK without Labels Π_{NIZK}. The last ingredient for our ARS is a NIZK for the opening relation R_{open}, which in our instantiation is

$$R_{\mathsf{open}} = \{((\mathsf{pk}, \mathsf{ct} = (E_1, E_2), \mathsf{M}), \mathsf{sk}) \mid \mathsf{sk} \star E_0 = \mathsf{pk} \wedge \mathsf{M} \star \mathsf{sk} \star E_1 = E_2\} \ .$$

A sigma protocol for this relation was introduced in [EKP20, Sec. 3.2]. We can then turn this sigma protocol into an NIZK by applying the Fiat-Shamir transform. (Note that we do not need this NIZK to be online-extractable.)

Concrete Instantiation for Tab. 1. For our isogeny based instantiation, we chose an HIG and a PKE based on the CSIDH-512 group action. The structure of this class group has been computed [BKV19], which allows for more efficient proofs. We chose the challenge space as string of length $M = 855$ with Hamming weight $K = 19$. Most of the signature is independent of N, and contains a fixed number of curves and class group elements as well as some overhead from the generic construction such as a hash value, the internal nodes in the seed tree, and commitment randomness to open the commitments. The only reason the signature size increases with N is that the signature contains a fixed amount of paths in a Merkle tree of depth $\log_2 N$. This makes for a very mild dependence on N.

7.2 Instantiation from Lattices

We instantiate a group-action-based HIG and PKE, and the corresponding NIZKs for the relations R_{sig} and R_{open} based on lattices under the MSIS and MLWE assumptions. The choices for the integer n, modulus q, and ring R_q are provided in Sec. 2.6.

Group-Action-Based HIG. By Def. 4.1, it suffices to define the public parameter $\mathsf{pp}_1 = (G, S_1, S_2, \delta_x, X_0, \mathcal{X}, \star)$ generated by $\mathsf{RelSetup}$ and to check that the output of IGen defines a hard relation. The public parameters pp are defined as follows:

- $(G, \mathcal{X}) = (R_q^\ell \times R_q^k, R_q^k)$, where X_0 is an arbitrary element in $\mathcal{X}$,
- For $b \in \{0, 1\}$, $S_b = \{(\mathbf{s}, \mathbf{z}) \in G \mid \|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leq B_b\}$, where B_1, B_2 are positive integers such that $B_1 < B_2 < q$,
- $\delta_x = \left(\frac{2(B_2 - B_1)+1}{2B_2+1}\right)^{n(k+\ell)}$,
- The group action $\star : G \times \mathcal{X} \to \mathcal{X}$ is defined as $(\mathbf{s}, \mathbf{e}) \star \mathbf{w} = (\mathbf{A}\mathbf{s} + \mathbf{z}) + \mathbf{w}$, where $\mathbf{A} \in R_q^{k \times \ell}$ is a fixed matrix sampled uniformly by $\mathsf{RelSetup}$.

We define S_3 to be a subset of G with coefficients all bounded by $B_2 - B_1$. It can be checked that pp satisfies all the conditions in Def. 4.1, where δ_x follows by simply counting the points included in S_2 and S_3. It remains to check that the relation $\widetilde{R}_{\mathsf{pp}} = \{(\mathbf{b}, (\mathbf{s}, \mathbf{z})) \mid \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \wedge (\mathbf{s}, \mathbf{e}) \in S_2 + S_3\}$ defines a hard relation as defined in Sec. 3.1, where $S_2 + S_3$. Note that if the adversary $\mathcal{A}$ is restricted to output a witness $(\mathbf{s}, \mathbf{e}) \in S_1$, then this follows directly from the MLWE_{n,q,B_1} assumption. For our application, we have to further consider the scenario where $\mathcal{A}$ may output a witness $(\mathbf{s}, \mathbf{e})$ outside of S_1. We need to consider this case since our online extractor for the NIZK can only extract a witness in the relaxed relation $\widetilde{R}_{\mathsf{pp}}$ rather than R_{pp}.

The hardness of our group-action-based HIG follows naturally from the $\mathsf{MSIS}_{n,q,k,\ell,2B_2}$ and $\mathsf{sMLWE}_{n,q,k,\ell,B_1}$ assumptions. We only focus on an adversary $\mathcal{A}$ that outputs a witness $(\mathbf{s}, \mathbf{e})$ outside of S_1, since the other case simply follows from MLWE as we seen above. Let us construct an adversary $\mathcal{B}$ against the $\mathsf{MSIS}_{n,q,k,\ell,2B_2}$ problem by using $\mathcal{A}$ as a subroutine. $\mathcal{B}$, given $\mathbf{A}$ as input, samples a random $(\mathbf{s}, \mathbf{e}) \leftarrow S_1$, sets $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ and invokes $\mathcal{A}$ on input $\mathsf{pp}, \mathbf{b}$, where pp includes $\mathbf{A}$. When $\mathcal{A}$ outputs $(\mathbf{s}', \mathbf{e}')$, $\mathcal{B}$ submits $(\mathbf{s} + \mathbf{s}', \mathbf{e} + \mathbf{e}')$ as its solution. By assumption, $\|\mathbf{s} + \mathbf{s}'\|_\infty, \|\mathbf{e} + \mathbf{e}'\|_\infty \leq B_1 + B_2 + B_3 = 2B_2$ and they are non-zero. Therefore, $\mathcal{B}$ breaks the $\mathsf{MSIS}_{n,q,k,\ell,2B_2}$ problem as desired.

Finally, the same proof shows that our group-action-based HIG is a hard *multi-instance* generator based on the same assumptions.

Group-Action-Based PKE. We use a PKE scheme based on the Lindner-Peikert framework [LP11]. We first explain the public parameters $\mathsf{pp}_2 = (\overline{G}, \overline{G}_\mathsf{T}, \mathcal{Y}, \overline{S}_1, \overline{S}_2, \delta_y, D_\mathcal{Y}, \star_\mathsf{M}, \mathcal{M})$ generated by $\mathsf{PKE.Setup}$.[11]

- $(\overline{G}, \overline{G}_\mathsf{T}, \mathcal{Y}) = (R_q^k \times R_q^\ell \times R_q, R_q, R_q^k \times R_q)$,
- For $b \in \{0,1\}$, $\overline{S}_b = \{(\mathbf{r}, \mathbf{e}, e) \in \overline{G} \mid \|\mathbf{r}\|_\infty, \|\mathbf{e}\|_\infty, \|e\|_\infty \leq B_b\}$, where B_1, B_2 are positive integers such that $B_1 < B_2 < q$ and $4(nk+1)(2B_2 - B_1) \leq q$,
- $\delta_y = \left(\frac{2(B_2-B_1)+1}{2B_2+1}\right)^{n(k+\ell+1)}$,
- $D_\mathcal{Y}$ is a distribution that samples a uniform random $(\mathbf{A}, \mathbf{s}, \mathbf{z}) \in R^{k\times\ell} \times R_q^\ell \times R_q^k$ and outputs a group action $\star : \overline{G} \times \mathcal{Y} \to \mathcal{Y}$ defined as $(\mathbf{r}, \mathbf{e}, e) \star (\mathbf{w}, w) = ((\mathbf{A}^\top \mathbf{r} + \mathbf{e} + \mathbf{w}, \mathbf{b}^\top \mathbf{r} + e + w)$ and an element $Y = (\mathbf{w}, w) \in \mathcal{Y}$, where $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{z}$,
- $\star_\mathsf{M} : \overline{G}_\mathsf{T} \times \mathcal{Y} \to \mathcal{Y}$ is a group action defined as $\mathsf{M} \star_\mathsf{M} (\mathbf{c}, c) = (\mathbf{c}, c + \mathsf{M} \cdot \lfloor q/2 \rceil)$,
- The message space $\mathcal{M}$ is a subset of $\overline{G}_\mathsf{T} = R_q$ with coefficients in $\{0,1\}$.

We define S_3 to be a subset of G with coefficients all bounded by $B_2 - B_1$. It can be checked that pp satisfies the conditions in Def. 4.2, where δ_y follows by simply counting the points included in S_2 and S_3. The remaining algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ are defined as follows, where $U(B)$ denotes elements in R_q with infinity norm at most $B \in \mathbb{N}$:

$\mathsf{KeyGen}(\mathsf{pp})$: It samples a uniform random $(\mathbf{A}, \mathbf{s}, \mathbf{z}) \in R^{k\times\ell} \times U(B_1)^\ell \times U(B_1)^k$ and outputs $(\mathsf{pk}, \mathsf{sk}) = ((\star_\mathsf{pk}, \mathbf{0}), \mathbf{s})$, where $\mathbf{0}$ is the zero polynomial in $\mathcal{Y}$ and $\star_\mathsf{pk}$ is a group action defined as $(\mathbf{r}, \mathbf{e}, e) \star_\mathsf{pk} (\mathbf{w}, w) = ((\mathbf{A}^\top \mathbf{r} + \mathbf{e} + \mathbf{w}, \mathbf{b}^\top \mathbf{r} + e + w)$, where $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{z}$. Note that pk is distributed as a sample from $D_\mathcal{Y}$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{M})$: On input a public key $\mathsf{pk} = (\star_\mathsf{pk}, Y_\mathsf{pk} = \mathbf{0})$ and a message $\mathsf{M} \in \mathcal{M}$, it samples $(\mathbf{r}, \mathbf{e}, e) \leftarrow \overline{S}_1$ and returns $\mathsf{ct} = \mathsf{M} \star_\mathsf{M} ((\mathbf{r}, \mathbf{e}, e) \star_\mathsf{pk} \mathbf{0}) = (\mathbf{A}^\top \mathbf{r} + \mathbf{e}, \mathbf{b}^\top \mathbf{r} + e + \mathsf{M} \cdot \lfloor q/2 \rceil) \in \mathcal{Y}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$: It parses $(\mathbf{c}, c) \leftarrow \mathsf{ct}$ and computes $w = c - \mathbf{c}^\top \mathbf{s}$ over R_q. It rounds each coefficient back to either 0 or $\lfloor q/2 \rceil$ whichever is closest modulo q and outputs the polynomial.

Correctness is a consequence of $(\mathcal{R}', \mathcal{KR}')$-correctness, which we show below, and decryption efficiency clearly holds as well. We discuss the remaining properties.

Multi-Challenge $\mathsf{IND\text{-}CPA}$ *Security.* The security follows by a standard proof using dMLWE. For completeness, we provide the proof: We consider a sequence of games and prove that the adversary's advantage only changes negligibly in each adjacent games. The first game is the original security game. In the second game, we modify the group action $\star_\mathsf{pk}$ included in the public key to be defined by a random $(\mathbf{A}, \mathbf{b}) \leftarrow R^{k\times\ell} \times R_q^k$. By the $\mathsf{dMLWE}_{n,q,k,\ell,B_1}$ assumption, this game is indistinguishable from the previous game. In the final game, we sample each ciphertext as $\mathsf{ct} \leftarrow R^k \times R_q$. By the $\mathsf{dMLWE}_{n,q,\ell+1,k,B_1}$ assumption, this game is indistinguishable from the previous game. Note that we appropriately parse the matrix $\mathbf{A}' \in R_q^{(\ell+1)\times k}$ provided by the challenge as $\mathbf{A}$ and $\mathbf{b}$, and query the oracle once for each ciphertext. Since the challenge bit b is statistically hidden from the adversary, no adversary has advantage in winning this game. This concludes the proof.

We note that we can prove multi-challenge $\mathsf{IND\text{-}CPA}$ security while only relying on the dMLWE assumption with a fixed number of instances (i.e., those that do not rely on the number of challenge ciphertexts), if we can tolerate choosing slightly less efficient parameters. Specifically, we can use the dual-Regev encryption [GPV08], where $\mathbf{A}$ is a tall matrix. When $\mathbf{A}$ is tall enough, $\mathbf{A}^\top \mathbf{r}$ and $\mathbf{b}^\top \mathbf{r}$ is distributed statistically close to random under appropriate choices of parameters owing to the regularity lemma [LPR13]. Hence, we only need the dMLWE assumption to jump from the first to second game above.

$(\mathcal{R}', \mathcal{KR}')$*-correctness.* We define $\mathcal{R}'$ and $\mathcal{KR}'$ as follows, where the choice of $\mathcal{R}'$ coincides with those considered in Thm. 5.1:

---

[11] Note that although we use the same (q, B_1, B_2) as those used by the group-action-based HIG, they can be set differently. We only use the same notations for better readability.

- $(\mathcal{R}', \mathcal{KR}') = (\overline{S}_2 + \overline{S}_3, U(2B_2 - B_1)^\ell \times U(2B_2 - B_1)^k)$, where recall S_3 is a subset of G with ring elements whose coefficients are all bounded by $B_2 - B_1$. Specifically, $\overline{S}_2 + \overline{S}_3 = \{(\mathbf{r}, \mathbf{e}, e) \in \overline{G} \mid \|\mathbf{r}\|_\infty, \|\mathbf{e}\|_\infty, \|e\|_\infty \leq 2B_2 - B_1\}$.

We check that correctness holds even if the ciphertext is encrypted using randomness $(\mathbf{r}, \mathbf{e}, e) \in \mathcal{R}'$ and a secret key $\mathsf{sk} = (\mathbf{s}, \mathbf{e}) \in \mathcal{KR}'$. Let $\mathsf{ct} = (\mathbf{A}^\top \mathbf{r} + \mathbf{e}, \mathbf{b}^\top \mathbf{r} + e + \mathsf{M} \cdot \lfloor q/2 \rceil)$, then $c - \mathbf{c}^\top \mathbf{s} = \mathsf{M} \cdot \lfloor q/2 \rceil + e + \mathbf{e}^\top \mathbf{s} - \mathbf{z}^\top \mathbf{r}$. Then, $\|e + \mathbf{e}^\top \mathbf{s} - \mathbf{z}^\top \mathbf{r}\|_\infty \leq \|e\|_\infty + \|\mathbf{e}^\top \mathbf{s}\|_\infty + \|\mathbf{z}^\top \mathbf{r}\|_\infty \leq (2B_2 - B_1) + 2nk(2B_2 - B_1)^2 \leq q/4$, where the last inequality follows from our parameter choice. Thus, M can be correctly decrypted with probability 1.

Multi-Proof Online Extractable NIZK with Labels $\Pi_{\mathsf{NIZK,lbl}}$. Using the group-action-based HIG and PKE, we can instantiate $\Pi_{\mathsf{NIZK,lbl}}$ for the signing relations R_{sig} and R'_{sig} (see Sec. 3.1) as explained in Secs. 5 and 6.

Statistically Sound NIZK without Labels Π_{NIZK}. It remains to show how to construct Π_{NIZK} for the opening relations R_{open} and R'_{open}. We can rewrite the relation R_{open} (see Sec. 3.1) as follows:

$$R_{\mathsf{open}} = \left\{ ((\mathsf{pk} = \mathbf{b}, \mathsf{ct} = (\mathbf{c}, c), \mathsf{M}), \mathsf{sk} = (\mathbf{s}, \mathbf{z})) \;\middle|\; \begin{array}{c} \|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leq B_1 \wedge \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{z} \\ \wedge \; \|c - \mathbf{c}^\top \mathbf{s} - \mathsf{M} \cdot \lfloor q/2 \rceil\|_\infty \leq q/4 \end{array} \right\}.$$

Notice we can rewrite the righthand side as

$$\underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{c}^\top \end{bmatrix}}_{\tilde{\mathbf{A}}} \mathbf{s} + \underbrace{\begin{bmatrix} \mathbf{z} \\ 0 \end{bmatrix}}_{\tilde{\mathbf{z}}} = \underbrace{\begin{bmatrix} \mathbf{b} \\ c - \mathsf{M} \cdot \lfloor q/2 \rceil + d \end{bmatrix}}_{\tilde{\mathbf{b}}},$$

where d is some element in R_q such that $\|d\|_\infty \leq q/4$. Since d is not secret, we can think d is included in the statement $(\mathsf{pk}, \mathsf{ct}, \mathsf{M})$. Then, Π_{NIZK} can simply viewed as an NIZK for the standard MLWE-based statement $\tilde{\mathbf{A}}\mathbf{s} + \tilde{\mathbf{z}} = \tilde{\mathbf{b}}$, where $\|\mathbf{s}\|_\infty, \|\tilde{\mathbf{z}}\|_\infty \leq B_1$. Notice that such a statement is implicitly used in $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig} since this statement is essentially the group-action-based HIG. Specifically, if we remove all the components regarding the OR proof and leave the proof regarding the group-action-based HIG from Figs. 3, 4 and 7, we arrive at our desired NIZK. Similarly to $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig}, we can only prove that a cheating prover was using a witness (i.e., secret key) satisfying $\|\mathbf{s}\|_\infty, \|\tilde{\mathbf{z}}\|_\infty \leq B_2 + B_3$. This is exactly the $\mathcal{KR}'$ defined above and coincides with the relaxed relation $\tilde{R}_{\mathsf{open}}$.

One may wonder if we can construct an NIZK for this standard MLWE relation based on a sigma protocol with a non-binary challenge set. Although the proof size of Π_{NIZK} is already constant, this may further minimize the proof size of the opening proof. We claim that this may be difficult. The main reason is that when we use a non-binary challenge space, the extracted witness $(\mathbf{s}, \tilde{\mathbf{z}})$ typically comes from a *furthered* relaxed relation such that not only they have a larger norm, they are guaranteed to only satisfy $\tilde{\mathbf{A}}\mathbf{s} + \tilde{\mathbf{z}} = t \cdot \tilde{\mathbf{b}}$ for some short $t \in R_q$. This relaxation may suffice in some settings but it turns out that it won't for ours as we can no longer prove $(\mathcal{R}', \mathcal{KR}')$-correctness. When restricted to binary challenges, we can control t to be $1 \in R_q$.

Remark 7.1 (Bai-Galbraith Optimization [BG14]). *We can apply the Bai-Galbraith optimization [BG14] by exploiting the lattice structure. This is a common and simple optimization used in various lattice-based interactive protocols based on the Fiat-Shamir with aborts paradigm [Lyu12] that allows to roughly halve the proof size, or signature size when viewing the proof as a signature, with no additional cost. Intuitively, for* MLWE*, proving knowledge of a short* $\mathbf{s}$ *indirectly proves knowledge of a short* $\mathbf{e}$ *since it is uniquely defined as* $\mathbf{b} - \mathbf{A}\mathbf{s}$*. Therefore, we can remove the components that are used to explicitly prove that* $\mathbf{e}$ *is short. Since the size of* $\mathbf{s}$ *and* $\mathbf{e}$ *are about the same in our construction, this allows to almost halve the proof size. For further details, see for example [BG14, DKL⁺18, BKP20].*

Concrete Instantiation for Tab. 1. For the concrete instantiation in Tab. 1, we use $M = 1749, K = 16$. For the HIG, we chose the parameters according to the parameters used in the Security Level II variant of the (round 3) NIST submission of the Dilithium signature scheme. Concretely, we use the ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, with $n = 256$ and $q = 2^{23} - 2^{13} + 1$, and we put $l = k = 4$, $B_1 = 2, B_2 = 2^{17}$. These parameters are chosen by the Dilithium team such that the relevant MLWE and MSIS problems are hard enough to reach NIST SL II.

For the PKE, we use the ring R'_q with $n = 256$ and $q' \approx 2^{49}$, and we put $k = l = 8, B_1 = 1, B_2 \approx 2^{16.3}$. The LWE estimator of Albrecht et al. estimates that this MLWE instance has 141 bits of security [APS15]. Moreover, the $(\mathcal{R}', \mathcal{KR}')$-correctness holds, because we have $(2B_2 - B_1) + 2nk(2B_2 - B_1)^2 \leq q/4$. For the parameter set without manager accountability, we only require $(\mathcal{R}', \mathcal{KR})$-correctness, so we only need $(2B_2 - B_1) + 2nk(2B_2 - B_1)B_1 \leq q/4$. Therefore, we can choose our parameters as $q' \approx 2^{30}, l = k = 5, B_1 = 1$, and $B_2 = 2^{15.9}$ for better signature sizes. The LWE estimator of Albrecht et al. estimates that this MLWE instance has also 141 bits of security. In either cases, we use an optimization due to Bai and Galbraith to reduce the size of the proofs (and therefore the size of the signature).

Similar to the isogeny instantiation, the signature size depends very mildly on N because N only affects the length of some paths in the signature. Finally, we can use Sec. 5.3 to obtain a tightly secure scheme. Since $K = 16$, the overhead compared to the non-tight scheme is a mere 512B.

Acknowledgements

Yi-Fu Lai was supported by the Ministry for Business, Innovation and Employment in New Zealand. Shuichi Katsumata was supported by JST CREST Grant Number JPMJCR19F6, Japan. This work was supported by CyberSecurity Research Flanders with reference number VR20192203, and in part by the Research Council KU Leuven grant C14/18/067 on Cryptanalysis of post-quantum cryptography. Ward Beullens is funded by FWO Junior Postdoc- toral Fellowship 1S95620N.

References

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 351–367. Springer, Heidelberg, September 2018.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 470–499. Springer, Heidelberg, August 2020.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2002.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `https://eprint.iacr.org/2015/046`.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 243–265. Springer, Heidelberg, September 2015.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136. Springer, Heidelberg, June 2016.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 551–572. Springer, Heidelberg, December 2014.

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 381–398. Springer, Heidelberg, September 2010.

[BCN18] Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 163–182. Springer, Heidelberg, July 2018.

[BFW15] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 629–649. Springer, Heidelberg, March / April 2015.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, Heidelberg, February 2014.

[BHSB19] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2181–2198. ACM Press, November 2019.

[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, Heidelberg, December 2020.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[BL07] Ernie Brickell and Jiangtao Li. Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 21–30, 2007.

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 409–441. Springer, Heidelberg, May 2019.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202. Springer, Heidelberg, August 2019.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.

[Cam97] Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 465–479. Springer, Heidelberg, May 1997.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer, Heidelberg, December 2000.

[CHH$^+$21] Kai-Min Chung, Yao-Ching Hsieh, Mi-Ying Huang, Yu-Hsuan Huang, Tanja Lange, and Bo-Yin Yang. Group signatures and accountable ring signatures from isogeny-based assumptions. Cryptology ePrint Archive, Report 2021/1368, 2021. https://ia.cr/2021/1368.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

[CLM$^+$18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003.

[CS20] Remi Clarisse and Olivier Sanders. Group signature without random oracles from randomizable signatures. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020*, volume 12505 of *LNCS*, pages 3–23. Springer, Heidelberg, November / December 2020.

[CSCDJRH20] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: Square-root vélu Quantum-resistant isogeny Action with Low Exponents. *IACR Cryptol. ePrint Arch.*, 2020:1520, 2020.

[CSV20] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional Diffie-Hellman problem for class group actions using genus theory. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 92–120. Springer, Heidelberg, August 2020.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/839`.

[dLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 574–591. ACM Press, October 2018.

[DN19] Itai Dinur and Niv Nadler. Multi-target attacks on the Picnic signature scheme and related protocols. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 699–727. Springer, Heidelberg, May 2019.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210. Springer, Heidelberg, September 2006.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.

[EK18] Ali El Kaafarani and Shuichi Katsumata. Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 89–119. Springer, Heidelberg, March 2018.

[EKP20] Ali El Kaafarani, Shuichi Katsumata, and Federico Pintore. Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 157–186. Springer, Heidelberg, May 2020.

[ELL$^+$15] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 260–285. Springer, Heidelberg, November / December 2015.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 259–288. Springer, Heidelberg, December 2020.

[ESLL19] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Heidelberg, August 2019.

[ESZ22] Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. Matrict+: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545, To appear in IEEE S&P 2022.

[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.

[FHJ20] Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-OR proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 212–244. Springer, Heidelberg, May 2020.

[FI06] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 89(5):1328–1338, 2006.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, August 1997.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

[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, Heidelberg, August 1987.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 395–412. Springer, Heidelberg, December 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Heidelberg, December 2007.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KP17] Sudhakar Kumawat and Souradyuti Paul. A new constant-size accountable ring signature scheme without random oracles. In *International Conference on Information Security and Cryptology*, pages 157–179. Springer, 2017.

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[Kup13] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*, volume 22, pages 20–34, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003.

[KY19] Shuichi Katsumata and Shota Yamada. Group signatures without NIZK: From lattices in the standard model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 312–344. Springer, Heidelberg, May 2019.

[LD21] Yi-Fu Lai and Samuel Dobson. Collusion resistant revocable ring signatures and group signatures from hard homogeneous spaces. Cryptology ePrint Archive, Report 2021/1365, 2021. `https://ia.cr/2021/1365`.

[LGd21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 213–241. Springer, Heidelberg, October 2021.

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 41–61. Springer, Heidelberg, December 2013.

[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, Heidelberg, May 2016.

[LMPY16] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical "signatures with efficient protocols" from simple assumptions. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 511–522. ACM Press, May / June 2016.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1051–1070. ACM Press, November 2020.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. SMILE: Set membership from ideal lattices with applications to ring signatures and confidential transactions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 611–640. Springer, Heidelberg, August 2021.

[LNWX18] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 58–88. Springer, Heidelberg, March 2018.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.

[LPY15] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, Heidelberg, August 2015.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

[LZCS16] Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient sanitizable signatures without random oracles. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part I*, volume 9878 of *LNCS*, pages 363–380. Springer, Heidelberg, September 2016.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[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, Heidelberg, December 2001.

[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, Heidelberg, April 2015.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pages 271–286. Springer, 2004.

[YAZ$^+$19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 147–175. Springer, Heidelberg, August 2019.

A Omitted Primitives

A.1 Index-hiding Merkle trees

The definition an *index-hiding* Merkle tree is taken almost verbatim from [BKP20]. Merkle trees [Mer88] allow one to hash a list of elements $A = (a_0, \cdots, a_N)$ into one hash value (often called the root). At a later point, one can efficiently prove to a third party that an element a_i was included at a certain position in the list A. In the following, we consider a slight modification of the standard Merkle tree construction, such that one can prove that a single element a_i was included in the tree without revealing its position in the list. Formally, the Merkle tree technique consists of three algorithms ($\mathsf{MerkleTree}$, $\mathsf{getMerklePath}$, $\mathsf{ReconstructRoot}$) with access to a common hash function $\mathcal{H}_{\mathsf{Coll}} : \{0,1\}^\star \to \{0,1\}^{2\lambda}$.

- $\mathsf{MerkleTree}(A) \to (\mathsf{root}, \mathsf{tree})$: On input a list of 2^k elements $A = (a_1, \cdots, a_{2^k})$, with $k \in \mathbb{N}$, it constructs a binary tree of height k with $\{l_i = \mathcal{H}_{\mathsf{Coll}}(a_i)\}_{i \in [2^k]}$ as its leaf nodes, and where every internal node h with children h_{left} and h_{right} equals the hash digest of a concatenation of its two children. While it is standard to consider the concatenation $h_{\mathsf{left}} \parallel h_{\mathsf{right}}$, we consider a variation which consists in ordering the two children according to the lexicographical order (or any other total order on binary strings). We denote by $(h_{\mathsf{left}}, h_{\mathsf{right}})_{\mathsf{lex}}$ this modified concatenation. The algorithm then outputs the root root of the Merkle tree, as well as a description of the entire tree tree.
- $\mathsf{getMerklePath}(\mathsf{tree}, I) \to \mathsf{path}$: On input the description of a Merkle tree tree and an index $i \in [2^k]$, it outputs the list path, which contains the sibling of l_i (i.e. a node, different from l_i, that has the same parent as l_i), as well as the sibling of any ancestor of l_i, ordered by decreasing height.
- $\mathsf{ReconstructRoot}(a, \mathsf{path}) \to \mathsf{root}$: On input an element a in the list of elements $A = (a_1, \cdots, a_{2^k})$ and $\mathsf{path} = (n_1, \cdots, n_k)$, it outputs a reconstructed root $\mathsf{root}' = h_k$, which is calculated by putting $h_0 = \mathcal{H}_{\mathsf{Coll}}(a)$ and defining h_i for $i \in [k]$ recursively as $h_i = \mathcal{H}_{\mathsf{Coll}}((h_{i-1}, n_i)_{\mathsf{lex}})$.

If the hash function $\mathcal{H}_{\mathsf{Coll}}$ that is used in the Merkle tree is collision-resistant, then the following easy lemma implies that the Merkle tree construction is *binding*, i.e. that one cannot construct a path that "proves" that a value $b \notin A = (a_1, \ldots, a_N)$ is part of the list A that was used to construct the Merkle tree without breaking the collision-resistance of the underlying hash function $\mathcal{H}_{\mathsf{Coll}}$.

Lemma A.1 (Binding for Merkle Tree)**.** *There is an efficient extractor algorithm that, given the description* tree *of a Merkle tree (having root* root *and constructed using the list of elements A) and* (b, path) *such that* $b \notin A$ *and* $\mathsf{ReconstructRoot}(b, \mathsf{path}) = \mathsf{root}$*, outputs a collision for the hash function* $\mathcal{H}_{\mathsf{Coll}}$.

The use of the lexicographical order to concatenate two children nodes in the Merkle tree construction implies that the output path of the $\mathsf{getMerklePath}$ algorithm information-theoretically hides the index $i \in [N]$ given as input. Formally, we have the following.

Lemma A.2 (Index Hiding for Merkle Tree)**.** *Let $N \in \mathbb{N}$ be a power of 2, D, D' be two arbitrary distributions over $\{0,1\}^*$ and D_I, with $I \in [N]$, be the distribution defined as*

$$D_I = \left[(a_I, \mathsf{path}, \mathsf{root}) \;\middle|\; \begin{array}{c} a_I \leftarrow D, \\ a_i \leftarrow D' \quad \forall\, 1 \le i \neq I \le N, \\ (\mathsf{tree}, \mathsf{root}) \leftarrow \mathsf{MerkleTree}(A), \\ \mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I) \end{array} \right]$$

where $A = (a_1, \ldots, a_N)$. Then we have $D_I = D_J$ for all $I, J \in [N]$.

A.2 Seed Tree

The definition seed tree is taken almost verbatim from [BKP20]. The purpose of a seed tree is to first generate a number of pseudorandom values and later disclose an arbitrary subset of them, without revealing information on the remaining values. The seed tree is a complete binary tree[12] of λ-bit seed values such that the left (resp. right) child of a seed seed_h is the left (resp. right) half of $\mathsf{Expand}(\mathsf{seed}\|h)$, where Expand is a pseudorandom generator (PRG). The unique identifier h of the parent seed is appended to separate the input domains of the different calls to the PRG. A sender can efficiently reveal the seed values associated with a subset of the set of leaves by revealing the appropriate set of internal seeds in the tree. We provide the full detail of the seed tree below. Let $\mathsf{Expand} : \{0,1\}^{\lambda+\lceil\log_2(M-1)\rceil} \to \{0,1\}^{2\lambda}$ be a PRG for any $\lambda, M \in \mathbb{N}$, instantiated by a random oracle $\mathcal{O}$. Then, a seed tree consists of the following four oracle-calling algorithms.

- $\mathsf{SeedTree}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, M) \to \{\mathsf{leaf}_i\}_{i\in[M]}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^{\lambda}$ and an integer $M \in \mathbb{N}$, it constructs a complete binary tree with M leaves by recursively expanding each seed to obtain its children seeds. Calls are of the form $\mathcal{O}(\mathsf{Expand}\|\mathsf{seed}_h\|h)$, where $h \in [M-1]$ is a unique identifier for the position of seed in the binary tree.
- $\mathsf{ReleaseSeeds}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}) \to \mathsf{seeds}_{\mathsf{internal}}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^{\lambda}$, and a challenge $\mathbf{c} \in \{0,1\}^M$, it outputs the list of seeds $\mathsf{seeds}_{\mathsf{internal}}$ that covers all the leaves with index i such that $c_i = 1$. Here, we say that a set of nodes D *covers* a set of leaves S if the union of the leaves of the subtrees rooted at each node $v \in D$ is exactly the set S.
- $\mathsf{RecoverLeaves}^{\mathcal{O}}(\mathsf{seeds}_{\mathsf{internal}}, \mathbf{c}) \to \{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=1}$: On input a set $\mathsf{seeds}_{\mathsf{internal}}$ and a challenge $\mathbf{c} \in \{0,1\}^M$, it computes and outputs all the leaves of subtrees rooted at seeds in $\mathsf{seeds}_{\mathsf{internal}}$. By construction, this is exactly the set $\{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=1}$.
- $\mathsf{SimulateSeeds}^{\mathcal{O}}(\mathbf{c}) \to \mathsf{seeds}_{\mathsf{internal}}$: On input a challenge $\mathbf{c} \in \{0,1\}^M$, it computes the set of nodes covering the leaves with index i such that $c_i = 1$. It then randomly samples a seed from $\{0,1\}^{\lambda}$ for each of these nodes, and finally outputs the set of these seeds as $\mathsf{seeds}_{\mathsf{internal}}$.

By construction, the leaves $\{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=1}$ output by $\mathsf{SeedTree}(\mathsf{seed}_{\mathsf{root}}, M)$ are the same as those output by $\mathsf{RecoverLeaves}(\mathsf{ReleaseSeeds}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}), \mathbf{c})$ for any $\mathbf{c} \in \{0,1\}^M$. The last algorithm $\mathsf{SimulateSeeds}$ can be used to argue that the seeds associated with all the leaves with index i such that $c_i = 0$ are indistinguishable from uniformly random values for a recipient that is only given $\mathsf{seeds}_{\mathsf{internal}}$ and $\mathbf{c}$. More formally, we have the following.

Lemma A.3. *Fix any $M \in \mathbb{N}$ and any $\mathbf{c} \in \{0,1\}^M$. If we model Expand as a random oracle $\mathcal{O}$, then any (computationally unbounded) adversary $A^{\mathcal{O}}$ that makes Q queries to the random oracle $\mathcal{O}$ can distinguish the following two distributions D_1 and D_2 with distinguishing advantage bounded by $\frac{Q}{2^\lambda}$:*

$$D_1 : \left\{ \mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=0} \;\middle|\; \begin{array}{c} \mathsf{seed}_{\mathsf{root}} \leftarrow \{0,1\}^{\lambda} \\ \{\mathsf{leaf}_i\}_{i\in[M]} \leftarrow \mathsf{SeedTree}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, M) \\ \mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{ReleaseSeeds}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}) \end{array} \right\}$$

$$D_2 : \left\{ \mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=0} \;\middle|\; \begin{array}{c} \forall i \text{ s.t. } c_i = 0 : \mathsf{leaf}_i \leftarrow \{0,1\}^{\lambda} \\ \mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{SimulateSeeds}^{\mathcal{O}}(\mathbf{c}) \end{array} \right\}$$

Here, the distributions take into account the randomness used by the random oracle as well.

B Dynamic Group Signatures from Accountable Ring Signatures

In this section, we review briefly the definition of group signatures and explain how accountable ring signatures can be naturally viewed as group signatures. A formal treatment can be found in Bootle et al. [BCC+16]

---

[12] A *complete* binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

B.1 Preliminaries on Group Signatures

Group signatures can be divided into two primary types: static schemes [BMW03] and dynamic schemes [BSZ05]. Roughly, while static group signature require the group to be fixed at setup, dynamic group signatures allow members to join and leave the group at any time. This joining and leaving is administered by the group manager, who has the power to add and revoke membership — as well as the ability to revoke anonymity and reveal the specific signer of a certain signature. For a dynamic group signature scheme, the revocation mechanism can be handled by a separate entity called opening or tracing authority to offer better flexibility in the scheme and this makes only little difference regarding the security notions.

Informally, a dynamic group signature scheme consists of a setup algorithm Setup, key generation algorithms MKGen and UKGen for the group manager and group members (or users) respectively, and Sign, Verify, Open, and Judge algorithms which are counterparts of the ARS scheme functions of the same names. Additionally, an interactive Join protocol run between the group manager and a user allows users to be added to the group, while an $\mathsf{UpdateGroup}$ function allows the group manager to revoke a user's membership in the group dynamically (this is done via some publicly-published *group info* info).

Dynamic group signature schemes should satisfy standard security properties of correctness, anonymity, traceability and non-frameability [BSZ05, BCC+16]. Correctness ensures that a signature produced by a user running Sign after joining the group via Join is accepted by Verify. The inclusion of the Join function in this definition ensures joining works as intended, beyond just guaranteeing the signing algorithms's correctness. Full CCA-anonymity (often refereed simply as *full* anonymity) states that even under full key exposure of all group members (other than the group manager, who can trivially revoke anonymity via Open), and with access to an opening oracle, the user who generated a certain signature cannot be identified. More specifically, an adversary should be unable to distinguish between signatures generated by any two members of the adversary's choice— even if the adversary knows all secret keys involved. This notion is almost identical to its namesake in the ARS setting (Sec. 2.4). In contrast, CPA-anonymity is a weaker notion which still allows the adversary to learn all group members' keys, but removes access to the opening oracle. Weaker variants of these two are *selfless* CCA-anonymity and *selfless* CPA-anonymity where the adversary cannot obtain any secret keys of targeted members in the anonymity game. Traceability states that an adversary who is able to corrupt any members is not able to produce a signature for which Open fails to return an active member of the group even if the group manager's secret key is leaked. Finally, non-frameability states that even if the group manager and all but one of the group members are corrupted, they cannot forge or falsely attribute a signature to an honest member who did not produce it. These properties also imply what is usually called *unforgeability*, because if an adversary could produce a signature for a group they knew no secret keys for, the signature must either fail to Open to an active user, or would frame an honest member of the group—violating either traceability or non-frameability. We also remark a difference, usually being neglected, that the group manager can be corrupted in the security model of a dynamic group while a static variant only takes into account the exposure of the opening secret key [BMW03]. We refer the reader to [BCC+16] for more thorough definitions.

B.2 Constructing Group Signatures from ARS

For completeness, we now review the generic construction of a dynamic group signature scheme from an accountable ring signature scheme, by Bootle et al. [BCC+15, BCC+16]. Let Π_{ARS} be a secure ARS scheme, then we define a group signature scheme Π_{GS} as follows:

Let the group manager be the opening authority of Π_{ARS}, and let the group manager's keypair be $(\mathsf{gmpk} = \mathsf{opk}, \mathsf{gmsk} = \mathsf{osk})$. The group public key gpk is then set to $(\mathsf{gmpk}, \mathsf{pp})$, where pp is the output of $\mathsf{GS.Setup} := \mathsf{ARS.Setup}$. Define $\mathsf{GS.UKGen} := \mathsf{ARS.UKGen}$, so that users generate their own keypairs directly. The Join protocol proceeds by a user submitting their public key pk to the group manager, who appends it to the list of keys in $\mathsf{info}_\tau := [\mathsf{vk}_0, \dots, \mathsf{vk}_i]$ (the group info at epoch τ) and publishes $\mathsf{info}_{\tau+1}$. Membership is similarly revoked by the group manager via $\mathsf{UpdateGroup}$ by removing the user's public key from info_τ and publishing the updated info. Finally, define:

- $\mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{info}_\tau, \mathsf{sk}_i, \mathsf{M}) := \mathsf{ARS.Sign}(\mathsf{gmpk}, \mathsf{sk}_i, \mathsf{info}_\tau, \mathsf{M})$.

- $\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{info}_\tau, \mathsf{M}, \sigma) := \mathsf{ARS.Verify}(\mathsf{gmpk}, \mathsf{info}_\tau, \mathsf{M}, \sigma)$.
- $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{info}_tau, \mathsf{gmsk}, \mathsf{M}, \sigma)$ calls $(\mathsf{vk}_j, \pi) \leftarrow \mathsf{ARS.Open}(\mathsf{gmsk}, \mathsf{info}_\tau, \mathsf{M}, \sigma)$ and returns (j, π).
- $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{info}_\tau, \mathsf{M}, \sigma, (j, \pi)) := \mathsf{ARS.Judge}(\mathsf{gmpk}, \mathsf{info}_\tau, \mathsf{vk}_j, \mathsf{M}, \sigma, \pi)$.

Note that info_τ defines the ring of signers at epoch τ and should be publicly accessible, as too should be the index-to-public-key ($j \leftrightarrow \mathsf{vk}_j$) correspondence table, maintained by the group manager. As shown in [BCC+16], this generic construction of a group signature from an ARS is *tightly* secure assuming the ARS is secure. Hence, our ARS construction in Sec. 3.1 implies a secure dynamic group signature scheme. The type of security notions satisfied by the resulting group signature, e.g., full or selfless, CCA or CPA anonymity, is directly inherited from the ARS.

We note that this scheme's group info grows linearly in the number of group members. This is the same as all other proposed efficient post-quantum group signature constructions such as [EZS+19]. It remains an interesting open problem to construct a efficient group signature where the group info grows at most logarithmically in the number of group members.

Contents