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

Ward Beullens[1,2], Samuel Dobson[3], Shuichi Katsumata[4], Yi-Fu Lai[3], and Federico Pintore[5]

[1] imec-COSIC, KU Leuven, Belgium
[2] IBM Research, Zurich, Switzerland
wbe@zurich.ibm.com
[3] University of Auckland, New Zealand
samuel.dobson.nz@gmail.com ylai276@aucklanduni.ac.nz
[4] National Institute of Advanced Industrial Science and Technology (AIST), Japan
shuichi.katsumata@aist.go.jp
[5] Department of Mathematics, University of Bari, Italy
federico.pintore@uniba.it

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 [22], 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) [15] and enhanced privacy ID (EPID) [16].

Currently, there are versatile constructions of *efficient* group signatures from *classical* assumptions, e.g., [10,26,40,38,9,49,48,27,3,23]. 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 [4] 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 [39], there has been a rich line of subsequent works on lattice-based (and one code-based) group signatures, including but not limited to [45,33,47,50,41]. However, these results remained purely asymptotic. It was not until recently that efficient lattice-based group signatures appeared [14,25,32,31]. In [31], 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 [58,13,30,2,29,51,52], 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 [14,42,25,32,31] do not satisfy the ideal security properties (which are by now considered standard) formalized by Bootle et al. [11]. Thus, we are also interested in the following question:

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

Can we construct efficient post-quantum group signatures satisfying the ideal security properties formalized by Bootle et al. [11]?

To address these questions, in this work we focus on *accountable ring signatures* [57]. An accountable ring signature offers the flexibility of choosing the group of users when creating a signature (like a ring signature [55]), 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 [57,12,46,44,32], we advocate that they are as relevant and interesting as group and ring signatures. As shown by Bootle et al. [12], 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 [5] 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 $(N^2Q)^{-1} \cdot \epsilon^2$ 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 setting since only the smallest among the CSIDH parameters [20] enjoys properties suitable to achieve concrete efficiency [8].

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 [4] 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. [11] 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* [11], 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 [7], 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 $(N^2Q)^{-1} \cdot \epsilon^2$ as in prior works whose proofs rely on the forking lemma [34,54]. 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 [43] 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. [7] which have a loose security reduction.

Comparison to Prior Work. To the best of our knowledge, Esgin et al. [32,31] 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. [11]. Our scheme is the only one to achieve full CCA anonymity. Esgin et al. [31] 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 lose anonymity. Here, "full" means that the signing key of all the users may be exposed to the adversary. In contrast, Katz, Kolesnikov, and Wang [42] 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 [5,11] (illustrated in the "Manager Accountability"

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

	N					Hardness	Security	Anonymity	Manager
	2	2^5	2^6	2^{10}	2^{21}	Assumption	Level		Accountable
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
ESZ[31]	/	12	/	19	/	MSIS/MLWE	NIST 2	CPA	No
KKW[42]	/	/	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 [53].

column). The two schemes [42,31] assume the group manager honestly executes the opening algorithm and that everyone trusts the output. Put differently, a 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 [42], while larger compared to [31]. 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 $\mathsf{opk} = \perp$, 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 [17]. 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 [18,19] 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., [26,40,38,49,48,25], use an $\mathsf{IND\text{-}CCA}$ 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 $\mathsf{IND\text{-}CCA}$ 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 [24] 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 [37], 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* [35]. 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

[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 [6].

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., [28]. 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 [7] 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 [7] 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 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. [7] 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 [7] 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

[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.

the adversary. The OR-proof by [7], 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 [56]. 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 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. [7], 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 at 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 [34,54] 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 [43] 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 [1,36] 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 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.

[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.

2 Preliminaries

Due to page limitation, the notation we use is defined in the full version of the paper, as are the standard primitives such as relaxed sigma protocol in the random oracle model and PKE. 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 $\mathcal{O}$. 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. 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 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 [21]

Definition 1 (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:*

$\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W}) \to \pi/\bot :$ *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.*

$\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) \to \top/\bot :$ *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 omit the standard notions of correctness, zero-knowledge, and statistical soundness to the full version of the paper. Below, we define one of the core property we require from NIZK to construct our ARS.

Definition 2 (Multi-Proof Online Extractability). *A NIZK proof system Π_{NIZK} is (multi-proof) online extractable if there exists a PPT extractor* $\mathsf{OnlineExtract}$ *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 prover oracle and before submitting a proof to the extract oracle.*[7]
- $(\mathtt{prove}, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$*: 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 checks pass, 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* $\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins]$ *where the probability is also taken over the randomness used by the random oracle.*

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. When $\mathsf{L} = \{\perp\}$, then it is a standard NIZK.

2.2 Accountable Ring Signatures

We provide the definition of accountable ring signatures (ARSs), following the formalization introduced by Bootle et al. [12].

Definition 3 (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*.*

[7] 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.

$\mathsf{Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) \to \top/\bot$: *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 4 (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* $\mathsf{Adv}^{\mathsf{Correct}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$*.*

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 [4,11] 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 5 (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* $\mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = |\Pr[\mathcal{A} \textit{ wins}] - 1/2|$*.*

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 6 (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 $\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A}$ *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 [12].

Definition 7 (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 $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A}$ *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 8 (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 $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A}$ *wins*$]$.

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 [34,54]. 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 [43] 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 an 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 9 **($(\mathcal{R}', \mathcal{KR}')$-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* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) = m$*.*

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 an 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 $\Pi_{\mathsf{NIZK},\mathsf{lbl}} = (\mathsf{NIZK.Setup}_{\mathsf{lbl}}, \mathsf{NIZK.Prove}_{\mathsf{lbl}}, \mathsf{NIZK.Verify}_{\mathsf{lbl}})$ 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.

Due to page limitation, we refer to the full version of this paper for the correctness and security of our accountable ring signature scheme Π_{ARS}.

3.2 Accountable Ring Signature to Dynamic Group Signature

Accountable ring signatures are known to trivially imply dynamic group signatures [12,11]. A formal treatment is provided by Bootle et al. [11]. We remark that the transformation provided in [11] retains 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 the full version of this paper .

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

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

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

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

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

ARS.Judge(opk, R, vk, M, σ, π_open)
1: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
2: if $\nexists I : \mathsf{vk} = \mathsf{R}_I$ then
3:     return $\perp$.
4: $b_0 \leftarrow \mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma)$
5: $b_1 \leftarrow \mathsf{NIZK.Verify}((\mathsf{opk}, \mathsf{ct}, I), \pi_{\mathsf{open}})$
6: return $b_0 \wedge b_1$

ARS.Open(osk, R, M, σ)
1: if $\mathsf{ARS.Verify}(\mathsf{opk}, \mathsf{R}, \mathsf{M}, \sigma) = \perp$ then
2:     return $\perp$
3: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
4: $I \leftarrow \mathsf{Dec}(\mathsf{osk}, \mathsf{ct})$
5: $\pi_{\mathsf{open}} \leftarrow \mathsf{NIZK.Prove}((\mathsf{opk}, \mathsf{ct}, I), \mathsf{osk})$
6: 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.

3.3 Tightly Secure Variant

Observe the only source of loose reduction in the previous section was in the unforgeability proof (see the full version of this paper), where we assume each building blocks, i.e., NIZK and PKE, are tightly reduced to concrete hardness assumptions. In this subsection, we present a modification of the construction in Fig. 1 to obtain a tight reduction in the unforgeability proof by using the Katz-Wang method [43]. The main difference is that we rely on a multi-proof online extractable NIZK proof system with labels for the following family of relations:

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

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

We show in Sec. 5.3 that we can obtain such a NIZK efficiently by slightly tweaking our NIZK for R_{sig}. The high level idea of how to use the Katz-Wang technique along with our NIZK for $R_{\mathsf{sig}}^{\mathsf{Tight}}$ is provided in the technical overview. Due to page limitation, we provide the full detail in the full version of this paper.

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 10 (Group-Action-based Hard Instance Generator). *A group-action-based hard instance generator, $\mathsf{GA\text{-}HIG}$ in short, is a pair of efficient algorithms $(\mathsf{RelSetup}, \mathsf{IGen})$ with the following properties:*

- *On input a security parameter λ, $\mathsf{RelSetup}$ 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 $(s \star X_0, s) \in R_{\mathsf{pp}}$.*
- *$(\mathsf{RelSetup}, \mathsf{IGen})$ 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 11 (Group-action-based PKE). *A* group-action-based public-key encryption scheme*,* $\mathsf{GA\text{-}PKE}$ *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 12 (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 show 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 and $(\mathcal{R}', \mathcal{KR}')$-correct for some relaxed randomness set $\mathcal{R}'$ and some relaxed key relation $\mathcal{KR}'$ (Def. 9).

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 [34,54]. 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 [7] 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:

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

Recall that by definition of $\mathsf{GA\text{-}PKE}$ (Def. 11), 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

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

and builds a Merkle tree 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. 2. The full detail on its correctness and security is provided in the full version of this paper.

5.2 From Base to Main Traceable OR Sigma Protocol

In this section, we expand the challenge space of $\Pi_{\Sigma}^{\mathsf{base}}$ to make the soundness error negligibly small. Such expansion 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 [7] explained in the technical overview. As the way we apply these optimizations follows [7] closely, we leave the details of our main traceable OR sigma protocol, denoted by $\Pi_{\Sigma}^{\mathsf{tOR}}$, to the full version of this paper.

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).

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

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

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

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

Figure 2: 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.

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 (see the full version of this paper).

As explained in the technical overview, we can use the sigma protocol for R_{sig} along with the sequential OR-proof [36] 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. More details are provided in the full version of this paper.

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 $\Pi_{\mathsf{NIZK,lbl}}$. The construction of our $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig} is provide in Fig. 3.[8] 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 $C_{M,K}$.

```
Prove^O(lbl, ({X_i}_{i∈[N]}, pk, ct), (I, s_I, r))
 1: resp := ⊥
 2: while resp = ⊥ do
 3:     com ← P_1^O(({X_i}_{i∈[N]}, pk, ct), (I, s_I, r))
 4:     chall ← O(FS || lbl || ({X_i}_{i∈[N]}, pk, ct) || com)
 5:     resp ← P_2^O((I, s_I, r), chall)
 6: return π ← (com, chall, resp)

Verify^O(lbl, ({X_i}_{i∈[N]}, pk, ct), π)
 1: (com, chall, resp) ← π
 2: if accept ← V_2^O(com, chall, resp) ∧ chall = O(FS || lbl || ({X_i}_{i∈[N]}, pk, ct) || com)
    then
 3:     return ⊤
 4: else
 5:     return ⊥
```

Figure 3: A multi-proof online extractable NIZK with labels $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig} obtained by applying the Fiat-Shamir transform to the traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}} = (P = (P_1, P_2), V = (V_1, V_2))$ defined in the full version of this paper.

Correctness of $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig} follows directly from the correctness of the underlying traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{tOR}}$. We show in the full version of this paper that $\Pi_{\mathsf{NIZK,lbl}}$ is multi-proof online extractable and zero-knowledge. We highlight that while we show special soundness for $\Pi_{\Sigma}^{\mathsf{tOR}}$ with respect to the relaxed relation $\tilde{R}'_{\mathsf{sig}}$ (see the full version), $\Pi_{\mathsf{NIZK,lbl}}$ is multi-proof

[8] 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.

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. Due to page limitation, the proof of zero-knowledge is provided in the full version of the paper. Below, we provide the proof of the multi-proof online extractability. Formally, we have the following.

Theorem 13. *The* NIZK *with labels* $\Pi_{\mathsf{NIZK,lbl}}$ *in Fig. 3 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*

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

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

Proof. We begin the proof by providing the description of the online extractor 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 $j \in [M]$.[9]
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 $\left((\mathsf{salt} \,\|\, j \,\|\, \mathsf{Expand} \,\|\, \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\right) \in L_{\mathcal{O}}$, 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 $\mathsf{W} = \bot$.

[9] 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 $\Pi^{\mathsf{tOR}}_{\Sigma}$. We use resp_i without overlines to indicate elements of $\overline{\mathsf{resp}}$.

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. 2.

– 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 $\big((\mathsf{salt} \parallel j \parallel \mathsf{Expand} \parallel \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)\big) \in L_{\mathcal{O}}$, 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 the full version of this paper. 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. 14 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 $\Pi_{\Sigma}^{\mathsf{base}}$. Due to page limitation, the proof is provide in the full version of this paper.

Lemma 14. *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 the full version of the paper). Then, with probability at least $1 - T \cdot \big(Q_{\mathsf{salt}}/2^{2\lambda-1} + (M \cdot Q_{\mathsf{salt}})/2^{\lambda} + 1/\left|C_{M,K}\right|\big)$, 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.

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. 14, 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 $\mathcal{A}$. As shown in the full version of the paper, $\mathsf{OnlineExtract}$ 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 $\mathcal{O}(\mathsf{salt} \,\|\, j \,\|\, \mathsf{Coll} \,\|\, \cdot)$ or $\mathcal{O}(\mathsf{salt} \,\|\, j \,\|\, \mathsf{Com} \,\|\, \cdot)$ 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.

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. Specifically, we instantiate a group-action-based HIG and PKE, and the corresponding NIZKs for the relations R_{sig} and R_{open} from the CSIDH group action and the MLWE group action. We use well-known PKEs based on isogenies and lattices as the basis for the group-action-based PKE. Due to page limitation, the details are provided in the full version of this paper.

We finish by providing details on how we arrive at the concrete parameters presented in 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, 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.

For the lattice based instantiations, we use $M = 1749, K = 16$. Our HIG is based on the NIST security level 2 parameter set from the (Round 3) NIST submission Dilithium. Our PKE uses the Lindner-Peikert framework, where we are forced to use MLWE parameters with a large modulus ($q \approx 2^{49}$) to achieve the $(\mathcal{R}', \mathcal{KR}')$-correctness requirement. For the instantiation without manager accountability, we only need $(\mathcal{R}', \mathcal{KR})$-correctness which allows us to use smaller parameters ($q \approx 2^{30}$). 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. For precise parameters we refer to the full version of this paper. 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

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. *ASIACRYPT 2002*, pp. 415–432.
2. T. Attema, V. Lyubashevsky, and G. Seiler. Practical product proofs for lattice commitments. *CRYPTO 2020, Part II*, pp. 470–499.
3. M. Backes, L. Hanzlik, and J. Schneider-Bensch. Membership privacy for fully dynamic group signatures. *ACM CCS 2019*, pp. 2181–2198.
4. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003*, pp. 614–629.
5. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. *CT-RSA 2005*, pp. 136–153.
6. D. Bernhard, M. Fischlin, and B. Warinschi. Adaptive proofs of knowledge in the random oracle model. *PKC 2015*, pp. 629–649.
7. W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. *ASIACRYPT 2020, Part II*, pp. 464–492.
8. W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. *ASIACRYPT 2019, Part I*, pp. 227–247.
9. P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get shorty via group signatures without encryption. *SCN 10*, pp. 381–398.
10. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. *CRYPTO 2004*, pp. 41–55.
11. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. *ACNS 16*, pp. 117–136.
12. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short accountable ring signatures based on DDH. *ESORICS 2015, Part I*, pp. 243–265.
13. J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. *CRYPTO 2019, Part I*, pp. 176–202.
14. C. Boschini, J. Camenisch, and G. Neven. Floppy-sized group signatures from lattices. *ACNS 18*, pp. 163–182.

15. E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. *ACM CCS 2004*, pp. 132–145.
16. E. Brickell and J. 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*, pp. 21–30.
17. J. Camenisch. Efficient and generalized group signatures. *EUROCRYPT'97*, pp. 465–479.
18. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. *ASIACRYPT 2000*, pp. 331–345.
19. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *CRYPTO 2003*, pp. 126–144.
20. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: An efficient post-quantum commutative group action. *ASIACRYPT 2018, Part III*, pp. 395–427.
21. M. Chase and A. Lysyanskaya. On signatures of knowledge. *CRYPTO 2006*, pp. 78–96.
22. D. Chaum and E. van Heyst. Group signatures. *EUROCRYPT'91*, pp. 257–265.
23. R. Clarisse and O. Sanders. Group signature without random oracles from randomizable signatures. *ProvSec 2020*, pp. 3–23.
24. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *CRYPTO'98*, pp. 13–25.
25. R. del Pino, V. Lyubashevsky, and G. Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. *ACM CCS 2018*, pp. 574–591.
26. C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. *Progress in Cryptology - VIETCRYPT 06*, pp. 193–210.
27. D. Derler and D. Slamanig. Highly-efficient fully-anonymous dynamic group signatures. *ASIACCS 18*, pp. 551–565.
28. A. El Kaafarani, S. Katsumata, and F. Pintore. Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512. *PKC 2020, Part II*, pp. 157–186.
29. M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. *ASIACRYPT 2020, Part II*, pp. 259–288.
30. M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. *CRYPTO 2019, Part I*, pp. 115–146.
31. M. F. Esgin, R. Steinfeld, and R. K. Zhao. Matrict+: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545.
32. M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. *ACM CCS 2019*, pp. 567–584.
33. M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang. A provably secure group signature scheme from code-based assumptions. *ASIACRYPT 2015, Part I*, pp. 260–285.
34. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO'86*, pp. 186–194.
35. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. *CRYPTO 2005*, pp. 152–168.
36. M. Fischlin, P. Harasser, and C. Janson. Signatures from sequential-OR proofs. *EUROCRYPT 2020, Part III*, pp. 212–244.

37. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *CRYPTO'99*, pp. 537–554.
38. J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 89(5):1328–1338.
39. S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. *ASIACRYPT 2010*, pp. 395–412.
40. J. Groth. Fully anonymous group signatures without random oracles. *ASIACRYPT 2007*, pp. 164–180.
41. S. Katsumata and S. Yamada. Group signatures without NIZK: From lattices in the standard model. *EUROCRYPT 2019, Part III*, pp. 312–344.
42. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. *ACM CCS 2018*, pp. 525–537.
43. J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. *ACM CCS 2003*, pp. 155–164.
44. S. Kumawat and S. Paul. A new constant-size accountable ring signature scheme without random oracles. In *International Conference on Information Security and Cryptology*, pp. 157–179. Springer.
45. F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé. Lattice-based group signatures with logarithmic signature size. *ASIACRYPT 2013, Part II*, pp. 41–61.
46. R. W. F. Lai, T. Zhang, S. S. M. Chow, and D. Schröder. Efficient sanitizable signatures without random oracles. *ESORICS 2016, Part I*, pp. 363–380.
47. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. *EUROCRYPT 2016, Part II*, pp. 1–31.
48. B. Libert, F. Mouhartem, T. Peters, and M. Yung. Practical "signatures with efficient protocols" from simple assumptions. *ASIACCS 16*, pp. 511–522.
49. B. Libert, T. Peters, and M. Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. *CRYPTO 2015, Part II*, pp. 296–316.
50. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Constant-size group signatures from lattices. *PKC 2018, Part II*, pp. 58–88.
51. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Practical lattice-based zero-knowledge proofs for integer relations. *ACM CCS 2020*, pp. 1051–1070.
52. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. SMILE: Set membership from ideal lattices with applications to ring signatures and confidential transactions. *CRYPTO 2021, Part II*, pp. 611–640, Virtual Event, 2021.
53. C. Peikert. He gives C-sieves on the CSIDH. *EUROCRYPT 2020, Part II*, pp. 463–492.
54. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396.
55. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *ASIACRYPT 2001*, pp. 552–565.
56. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. *EUROCRYPT 2015, Part II*, pp. 755–784.
57. S. Xu and M. Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pp. 271–286. Springer, 2004.
58. R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. *CRYPTO 2019, Part I*, pp. 147–175.