Group Signatures and Accountable Ring Signatures from Isogeny-based Assumptions

Kai-Min Chung[1], Yao-Ching Hsieh[1], Mi-Ying Huang[2], Yu-Hsuan Huang[1], Tanja Lange[3], and Bo-Yin Yang[1]

[1] Academia Sinica, Taiwan
kmchung@iis.sinica.edu.tw, ychsieh@ntu.edu.tw, asd00012334.cs04@nctu.edu.tw, byyang@iis.sinica.edu.tw

[2] University of Southern California, United States
miying.huang@usc.edu

[3] Eindhoven University of Technology, the Netherlandss
tanja@hyperelliptic.org

Abstract. Group signatures are an important cryptographic primitive providing both anonymity and accountability to signatures. Accountable ring signatures combine features from both ring signatures and group signatures, and can be directly transformed to group signatures. While there exists extensive work on constructing group signatures from various post-quantum assumptions, there has not been any using isogeny-based assumptions. In this work, we propose the first construction of isogeny-based group signatures, which is a direct result of our isogeny-based accountable ring signature. This is also the first construction of accountable ring signatures based on post-quantum assumptions. Our schemes are based on the decisional CSIDH assumption (D-CSIDH) and are proven secure under the random oracle model (ROM).

1 Introduction

Group signatures, first proposed by Chaum and van Heyst [14], are signature schemes that permit signing by a *group*, a set of players chosen by a prescribed *group manager*. Each of the players can generate publicly verifiable signatures on behalf of the group while keeping itself anonymous to everyone except the group manager. The group manager has the authority to *open*, i.e. to reveal the signer's identity from a signature with its *master secret key*.

Since their proposal, there have been numerous works devoted to group signatures. Many of them aimed to give refinements and extensions to the primitive. Some of these extensions, such as separating the opening authority from the manager [3,11], or additionally requiring a "Judge" functionality to verify opening results [3], are widely adopted in the formulation for many succeeding works. Nevertheless, in this work, we will focus on the formulation which only contains fundamental components of a group signature.

Accountable Ring Signatures. One important line of research on group signatures studies variants with *dynamic groups*. In contrast to the original formulation where only *static groups* are supported [2,14], a dynamic group signature allows a group to be updated after the setup stage. The notion of *partially dynamic* group signatures was formulated by Bellare, Shi, and Zhang [3] and Kiayias and Yung [22], where parties can join a group but cannot be removed. There are also many works that achieve group signatures with removal of group members, as early as [9,34].

Accountable ring signatures (ARS), first proposed by Xu and Yung [38], provides the "dynamic property for groups" in a different aspect. ARS, while having a "ring signature" [32] within its name, can also be viewed as a variant of group signatures where groups are *fully dynamic* but *not authenticated*. In an ARS scheme, the manager no longer has any control over the group. Instead, a signer can freely decide which master public key to use and which group to sign for, and its identity can then be opened by the corresponding master secret key. Though seemingly incomparable to standard group signature, an ARS scheme can in fact trivially imply a group signature scheme, simply by fixing the group at the setup stage. Later, Bootle, Cerulli, Chaidos, Ghadafi, Groth, and Petit [8] proposed a stringent formulation for ARS, along with a provable construction based on the DDH assumption. It is further shown in [7] that such a stringent ARS scheme can be generally transformed to a *fully dynamic* group signature scheme.

Group Signatures from Post-quantum Assumptions. In the past decade, there has been increasing attention on the importance of post-quantum security for cryptographic primitives. Various attempts emerge to construct group signatures based on cryptographic assumptions that resist quantum attacks. Gordon, Katz, and Vaikuntanathan first gave a group signature construction from lattice-based assumptions [21]. It is followed by plenty of constructions of lattice-based group signatures, either for static groups [24, 29] or dynamic groups [26, 27]. There have also been a few attempts on constructing group signatures from other classes of post-quantum assumptions, such as code-based assumptions [18] or hash-based assumptions [1]. However, to the best of our knowledge, none of those are from *isogeny-based assumptions*, which is the main target of this work.

We also note that current constructions of accountable ring signatures are based on either pre-quantum assumptions (DDH [8], q-SDH [25]) or primitives in the absence of post-quantum constructions ($i\mathcal{O}$ [23]).

Our result. In this work, we construct a group signature scheme and an accountable ring signature scheme from *isogeny-based assumptions* in the *random oracle model* (ROM). To the best of our knowledge, this is the first construction of both isogeny-based group signatures and accountable ring signatures based on post-quantum assumptions.

We base our construction on the *decisional CSIDH assumption* (D-CSIDH). From an abstract viewpoint, D-CSIDH is a natural generalization of DDH which is built over the weaker group-action structure.

Due to the lack of the homomorphic property in group-action assumptions, it is usually infeasible to transform results obtained from group-based assumptions to those from group-action-based assumptions. Our work demonstrates the possibility for constructing advanced cryptographic primitives with group-action-based assumptions, despite its limited properties.

Our construction of ARS does not satisfy some of the (too) stringent definitions, such as the one from [8]. Nevertheless, it already suffices to imply a group signature. Therefore, we believe that our work would be a meaningful starting point for the study of advanced isogeny-based signatures.

1.1 Technical Overview

In this overview, we assume some familiarity for sigma protocols and the Fiat-Shamir transformation [20].

Signatures based on isogeny class group action. Stolbunov [35] gave a first attempt toward an isogeny-based signature scheme in his thesis. His scheme applies the Fiat-Shamir transformation [20] to the sigma protocol of Couveignes [15]. While Couveignes' protocol is structurally similar to the discrete log based protocol by Chaum and van Heyst [14], its challenge space cannot be extended as in Schnorr's protocol [33]. Parallel repetition is thus necessary for Stolbunov's signature scheme.

Later, following the proposal of an efficient class group action implementation by CSIDH [12], SeaSign [19] and CSI-FiSh [5] separately gave efficient signature constructions based on Stolbunov's approach. One main contribution of their works is that they overcome the lack of canonical representation for elements in the class group $\mathsf{Cl}(\mathcal{O})$. In Stolbunuov's scheme, the signer would reveal rs for $r \xleftarrow{\$} \mathsf{Cl}(\mathcal{O})$ and secret $s \in \mathsf{Cl}(\mathcal{O})$. However, since r and s are represented as element wise bounded vectors in the CSIDH representation, a naive representation for rs *does not* hide the information of s. To cope with this issue, SeaSign proposed a solution using the Fiat-Shamir with abort technique [28], while CSI-FiSh computes the whole class group structure and its relation lattice for a specific parameter set, CSIDH-512. In this work, we will adopt the latter approach, where we can simply assume canonical representation for elements in $\mathsf{Cl}(\mathcal{O})$.

Recently, Beullens, Katsumata, and Pintore [4] showed how to construct an isogeny-based ring signature with the sigma protocol for an OR-relation. Our work similarly starts with a sigma protocol which additionally supports an opening operation. We want a sigma protocol that takes n statements and a master public key as inputs, computes a proof for one of the statements and embeds the "identity" of the proved statement into the transcript so that it can be extracted with the master secret key. As the first step, we will discuss how we can embed information for opening into the transcript.

Embedding opening information. In a group signature scheme, the information for the signer's identity must be somehow embedded into the signature, so that the master can open it. One natural approach to embed opening information is to encrypt the information with the master public key. Such an approach is proven successful in a few previous works on group signatures [6,8]. However, since the opening information is now a ciphertext under the master key, a verifier could only check the validity of the ciphertext via homomorphic operations or NIZK. Unfortunately, unlike group-based assumptions, it is not yet known how to achieve such homomorphic property from the weaker *group action structure* given by isogeny-based assumptions. There is also no isogeny-based NIZK construction in the literature. Thus, we will have to come up with a structurally simpler way to encode our opening information.

In light of this, we construct our opening functionality in a very naive way. For a signature with group/ring size n and a master secret key s_m for opening, we embed the signer identity by one DDH tuple and $n-1$ dummies. Namely, the opening information is of the form

$$\tau = ((r_1E, r_2E, \ldots, r_nE), r_kE_m), \text{ where } r_1, \ldots, r_n \xleftarrow{\$} G \text{ and } E_m = s_mE \ ,$$

which embeds the signer's identity $k \in [n]$ through *position*, and is extractable for the manager holding s_m. Note that such τ keeps all its elements in the form of curves/set elements, hence the verifier can do further group action on τ for consistency checking. This circumvents the previous difficulty, but with the cost of a larger payload.

Openable Sigma Protocol. To construct a group signature/accountable ring signature scheme through Fiat-Shamir transformation, we first introduce an intermediate primitive called openable sigma protocol. We refer the reader to Section 3 for details.

The formulation of the openable sigma protocol looks similar to the standard OR sigma protocol. They both take n statements and one witness as input. However, there is a major difference between them. The OR sigma protocol is a proof of knowledge for the OR-relation. The openable sigma protocol, on the other hand, is a proof of knowledge for the relation of the kth statement, where k is chosen at the proving stage and embedded in the first message com, and could then be extracted by the master secret key s_m.

For our openable sigma protocol, the special soundness would thus require an extractor that extracts the kth witness which matches the opening result. Such a stronger extractor is crucial for proving unforgeability for group signatures, in which we transform a forger for party k into the extractor for the kth witness. Extractors for standard OR sigma protocols cannot provide such reduction.

Also, unlike an OR sigma protocol, an openable sigma protocol cannot get anonymity directly from the HVZK property, as the proving statement is now embedded in com. To achieve anonymity, we need an extra property *computa-*

tional witness indistinguishability (CWI) which states that, for an honest master key pair $(\mathsf{mpk}, \mathsf{msk})$, the proof for the k_1th statement is indistinguishable from the proof for the k_2th statement. This promises that when transformed to signatures, the signer would be anonymous as long as the manager has not colluded.

The construction of our openable sigma protocol is built on top of the previous identity embedding component. For statements $E_1 \dots, E_n$ along with the kth witness s_k s.t. $E_k = s_k E$ and the master key pair $(s_m, E_m = s_m E)$, the opening information in our protocol is set to

$$\tau = (E^\beta, E^{\mathsf{Open}}) = ((r_1 E_1, r_2 E_2, \dots, r_n E_n), r_k s_k E_m), r_1, \dots, r_n \overset{\$}{\leftarrow} G$$

As argued earlier, the manager can extract k from τ with s_m. To complete a proof of knowledge protocol, we use two challenges $(\mathsf{ch} = 1, 2)$ to extract the knowledge of each r_i, and use another two challenges $(\mathsf{ch} = 3, 4)$ to extract "some $d = r_k s_k$ s.t. $dE \in E^\beta$ and $dE_m = E^{\mathsf{Open}}$." This gives us a four challenge openable sigma protocol with corresponding special soundness property. We detail the full construction and the security proof in Section 3.

Soundness amplification and Fiat-Shamir Transformation. From our 4-challenge sigma protocol with opening property, we immediately obtain an identification scheme with soundness error $\frac{3}{4}$. It may be tempting to claim that we can achieve soundness $(\frac{3}{4})^\lambda$ through a λ repetition. Unfortunately, this is not the case because each parallel session can be independently generated with a different witness, and some of the witnesses might be validly owned by the adversary. As a concrete example, in a λ-parallel protocol, an adversary that owns 3 keys can generate $\lambda/4 - 1$ honest parallel sessions on behalf of each key, and then cheat on only $\lambda/4 + 3$ sessions to achieve a successful forgery. The succeeding probability is $(\frac{3}{4})^{\lambda/4+3}$ instead of $(\frac{3}{4})^\lambda$. Thus, for an adversary owning n_A keys, we would need $t = n_A(\lambda - 1) + 1 = O(n_A \lambda)$ repetition to ensure that, any adversary that wishes to successfully prove for an honest identity would need to cheat on at least λ parallel sessions, and the success probability is thus at most $(\frac{3}{4})^\lambda$.

With an identification scheme with negligible soundness error, we can now apply the Fiat-Shamir transformation and obtain a signature scheme. With the improved forking lemma [10] detailed in Section 2.4, a forging adversary A that can with non-negligible probability generate accepting tuples of $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ would imply an algorithm B that generates 4 accepting tuples with identical com and distinct ch. Note that as we are applying the forking lemma on a t-parallel protocol, obtaining 4 distinct challenges does not immediately imply extraction. For instance, the forking lemma may output 4 distinct vectors of length t, while on every position $i \in [t]$ the four vectors are not completely distinct, resulting in an extraction fail on every parallel session. Nevertheless, we can easily show that, under polynomial many rewinds on the random oracle, the probability of existence of such "bad" vector tuples among all queries is negligible. Thus, we can successfully extract witnesses with non-negligible probability, which immediately implies unforgeability.

2 Preliminary

2.1 Isogeny and Class Group Action

Here we first briefly cover the basics for *elliptic curve isogenies*. For simplicity, we consider a working (finite) field $\mathbb{F}_q$ with characteristic $p > 3$. An isogeny ϕ between elliptic curves $E_1 \to E_2$ defined over an algebraic closure $\bar{\mathbb{F}}_q$ is a surjective homomorphism between the groups of rational points $E_1(\bar{\mathbb{F}}_q) \to E_2(\bar{\mathbb{F}}_q)$ with a finite kernel. If, additionally, ϕ is assumed *separable*, i.e. the induced extension of function fields $\phi^* : \bar{\mathbb{F}}_q(E_2) \hookrightarrow \bar{\mathbb{F}}_q(E_1)$ by $\bar{\mathbb{F}}_p(E_2) \ni f \mapsto f \circ \phi \in \bar{\mathbb{F}}_p(E_1)$ is separable, then for any finite subgroup $H \leq E_1(\bar{\mathbb{F}}_p)$, there is an isogeny $\phi : E_1 \to E_2$ having H as its kernel, and the co-domain curve is furthermore uniquely determined up to isomorphisms (in $\bar{\mathbb{F}}_q$). We refer to the co-domain curve as the *quotient curve*, denoted E_1/H. A corresponding isogeny could be computed using Velu's formula specified in [37], which works by expanding the coordinates of $Q = \phi(P)$ as follows,

$$x(Q) = x(P) + \sum_{R \in H \setminus \{0\}} \left(x(P+R) - x(R) \right),$$
$$y(Q) = y(P) + \sum_{R \in H \setminus \{0\}} \left(y(P+R) - y(R) \right).$$

The separable degree $\deg_{\mathsf{sep}} \phi$ is defined as the separable degree for ϕ^*, which coincides with the size of its kernel $\# \ker \phi$, and since any isogeny could be acquired by precomposing Frobenius maps to a separable isogeny, i.e. of form $\phi \circ \pi_p^k$ where ϕ is separable, we can (equivalently) define the (full) degree $\deg\left(\phi \circ \pi_{p^k}\right) = \deg_{\mathsf{sep}}(\phi) p^k$. From now on, we will assume separability of isogenies unless otherwise specified, and therefore $\deg \phi = \deg_{\mathsf{sep}} \phi$ in this case.

For large degree ϕ, when both domain E_1 and co-domain E_2 (supersingular) curves are prescribed, it could be hard to determine the kernel (and thus ϕ). The current best-known (generic) quantum algorithm is *claw finding*, which takes $\tilde{O}\left(\deg(\phi)^{1/3}\right)$ operations.

One important structure for isogenies is the so-called *isogeny class group action*, which was first used for cryptographic constructions by [15, 35], and was viewed as a weaker alternative for discrete logarithm. However, although theoretically feasible, the instantiated group action used to rely heavily on techniques regarding the so-called *modular polynomials*, which is computationally expensive in practice. Later on, improvements in the *Commutative SIDH* (CSIDH) [12] scheme got rid of these techniques. Concretely, the space X is instantiated as a set $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p) = \{E/\mathbb{F}_p \text{ supersingular elliptic curves}\}/\cong_{\mathbb{F}_p}$ acted by their ideal class group $\mathsf{Cl}(\mathcal{O})$ of the $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O} = \mathsf{End}_p(E)$ where $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ but $\mathcal{O} \otimes \mathbb{Q}$ tensored as a $\mathbb{Z}$-module is identical regardless of the choice of $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ thus so is $\mathsf{Cl}(\mathcal{O})$. The additional parameter π_p denotes the p-power Frobenius $\pi_p : (x, y) \mapsto (x^p, y^p)$. Elements of $\mathsf{Cl}(\mathcal{O})$ are equivalence classes $\mathfrak{a}$ of ideals of the (partial) endomorphism ring $\mathcal{J} \lhd \mathsf{End}_p(\mathcal{O})$. Any such ideal class $\mathfrak{a} \in \mathsf{Cl}(\mathcal{O})$ therefore acts on the curves by sending $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ to the quotient curve $\mathfrak{a} \cdot E := E/E[\mathcal{J}]$ where $\mathcal{J} \in \mathfrak{a}$ is a representative of the equivalence class $\mathfrak{a}$ and $E[\mathcal{J}] = \bigcap_{f \in \mathcal{J}} \ker f$ is the simultaneous kernel of $\mathcal{J}$.

The working base field $\mathbb{F}_p$ for CSIDH is carefully selected such that $p = 4\ell_1 \cdots \ell_n - 1$ where each $\ell_i > 2$ is a small prime generally referred to as an *Elkies prime*. This allows one to generate a heuristically large enough sub-covering $\{\mathfrak{l}_1^{e_1} \dots \mathfrak{l}_n^{e_n} | \forall i : |e_i| \leq b_i\}$ of $\mathsf{Cl}(\mathcal{O})$ where each prescribed b_i is small[4] and each $\mathfrak{l}_i^{\pm 1}$ is the class of ideal $\langle \pi_p \mp 1, \ell_i \rangle$. The indices $(e_1, \dots, e_n)$ thus represent the ideal class $\mathfrak{l}_i^{e_1} \cdots \mathfrak{l}_n^{e_n}$, making it easier to compute the co-domain curve. In particular, for a curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ and any choice of ℓ_i, the curve $\mathfrak{l}_i \cdot E := E/E[\langle \pi_p - 1, \ell_i \rangle]$ is computed by sampling a generator of the kernel,

$$E[\langle \pi_p - 1, \ell_i \rangle] = E(\mathbb{F}_p)[\ell_i] = \{P \in E(\mathbb{F}_p) | \ell_i P = 0\},$$

which is a one dimensional $\mathbb{Z}/\ell$-linear eigen-subspace of π_p within the ℓ_i-torsion $E[\ell_i]$. For the opposite direction, one can compute $\mathfrak{l}_i^{-1} \cdot E = (\mathfrak{l}_i \cdot E^t)^t$ where the superscript ^{t} is referred to as the quadratic twist of the specified curve, by taking the convention that the curve is fixed when its j-invariant is 1728, or equivalently, this can be done by sampling from the other $\mathbb{Z}/\ell_i$-linear eigen-subspace of π_p in $E[\ell_i]$, which sits in the quadratic extension $E(\mathbb{F}_{p^2})$.

We also list here some well-known properties for the considered class group action. First, the class group $\mathsf{Cl}(\mathcal{O})$ *commutes*, which is a direct result of the fact that the $\mathbb{F}_p$-rational endomorphism ring $\mathsf{End}_p(E)$ commutes. Second, as noted in [12, Theorem 7], $\mathsf{Cl}(\mathcal{O})$ acts *freely and transitively* on $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$, which means that for all $E_1, E_2 \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$, there exists a unique $\mathfrak{a} \in \mathsf{Cl}(\mathcal{O})$ such that $\mathfrak{a} \cdot E_1 = E_2$. Finally, elements in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ can be *efficiently verified*. We note that a curve E is supersingular if and only if it has $p + 1$ points over $\mathbb{F}_p$. This can be efficiently tested by finding some $P \in E(\mathbb{F}_p)$ with order $\mathsf{ord}(P) \geq 4\sqrt{p}$ dividing $p+1$. A random point P sampled from $E(\mathbb{F}_p)$ satisfies such a condition with high probability if E is supersingular, and whether it does can be verified efficiently as follows. If $(p+1)P \neq 0$, then $\mathsf{ord}(P)$ does not divide $p+1$ and E is ordinary. Otherwise, we can perform the so-called *batch co-factor multiplication* computing $P_i = \frac{p+1}{\ell_i} P$ for each i, by using convention that $\ell_0 = 4$. This allows us to determine $\mathsf{ord}(P) = \prod_i \mathsf{ord}(P_i)$.

Hardness assumptions. Hardness for the *group action inverse problem* (GAIP) in Definition 1 is commonly assumed for the above-mentioned group action, which has been shown useful on constructing signature schemes such as CSI-FiSh [5] and SeaSign [19].

[4] For CSIDH-512 [12] proposes $b_1 = \cdots = b_n = 5$.

Definition 1. *(Group Action Inverse Problem (GAIP)) On inputs $E_1, E_2 \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$, find $\mathfrak{a} \in \mathsf{Cl}(\mathcal{O})$ such that $E_2 \cong_{\mathbb{F}_p} \mathfrak{a} \cdot E_1$.*

In this work, we need to assume hardness for a weaker problem, the *decisional CSIDH problem* (abbreviated as D-CSIDH[5]) in Definition 2, which was considered already in [15,35], and is the natural generalization of the decisional Diffie-Hellman problem for group actions.

Definition 2. *(Decisional CSIDH (D-CSIDH) / DDHAP) For $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$, distinguish the two distributions*

- $(E, \mathfrak{a}E, \mathfrak{b}E, \mathfrak{c}E)$, *where* $\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \xleftarrow{\$} \mathsf{Cl}(\mathcal{O})$,
- $(E, \mathfrak{a}E, \mathfrak{b}E, \mathfrak{a}\mathfrak{b}E)$, *where* $\mathfrak{a}, \mathfrak{b} \xleftarrow{\$} \mathsf{Cl}(\mathcal{O})$.

We note that for typical cryptographic constructions such as CSIDH, additional heuristic assumptions are required to sample a random element from the class group (as in Definition 2). This is because the "CSIDH-way" for doing this is by sampling exponents $(e_1, \ldots, e_n)$ satisfying $\forall i : |e_i| \leq b_i$, and the resulting distribution for ideals $\mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n}$ is generally non-uniform within $\mathsf{Cl}(\mathcal{O})$. To get rid of such heuristics, one could instead work with specific parameters, where a bijective (yet efficient) representation of ideals is known. For instance, in [5], the structure of $\mathsf{Cl}(\mathcal{O})$ is computed, including a full generating set of ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ and the entire lattice $\Lambda := \{(e_1, \ldots, e_n) | \mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n} = \mathsf{id}\}$. Evaluating the group action is just a matter of approximating a *closest vector* and then evaluating the residue as in CSIDH. In this work, we will be working with such a "perfect" representation of ideals, unless otherwise specified.

As a remark, we note that the D-CSIDH problem for characteristic $p = 1 \mod 4$ is known to be broken [13]. Nevertheless, the attack is not applicable to the standard CSIDH setting where $p = 3 \mod 4$.

2.2 Group Action DDH

In this section, we give an abstract version of the CSIDH group action. Such formulation will simplify our further construction and security proof.

A group action $\mathcal{GA}_\lambda = (G_\lambda, \mathcal{E}_\lambda)$ with security parameter λ (we will omit the subscripts for simplicity) is called a DDH-secure group action if the following holds:

- G acts freely and transitively on $\mathcal{E}$.
- DDHAP is hard on $\mathcal{GA}_\lambda$. i.e., for any efficient adversary A and $E \in \mathcal{E}$, the advantage for A distinguishing the following two distributions is $\mathsf{negl}(\lambda)$.

[5] This problem is called the decisional Diffie-Hellman group action problem (DDHAP) in [35].

- (E, aE, bE, cE), $a, b, c \xleftarrow{\$} G$
- (E, aE, bE, abE), $a, b \xleftarrow{\$} G$

As a side remark, the GAIP problem is also hard on a DDH-secure group action.

For a DDH-secure group action, we can also have a natural parallel extension for DDHAP. Such extension is also discussed in [17].

Definition 3. *(Parallelized-DDHAP (P-DDHAP)) Given $E \in \mathcal{E}$, distinguish the two distributions*

– $(aE, \{b_iE\}_{i\in[m]}, \{c_iE\}_{i\in[m]})$*, where* $a, \{b_i\}_{i\in[m]}, \{c_i\}_{i\in[m]} \xleftarrow{\$} G$*,*
– $(aE, \{b_iE\}_{i\in[m]}, \{ab_iE\}_{i\in[m]})$*, where* $a, \{b_i\}_{i\in[m]} \xleftarrow{\$} G$*.*

By a simple hybrid argument, we can easily see that if $DDHAP$ is ϵ-hard, then P-DDHAP is $m\epsilon$ hard. To see this, note that a single DDHAP can be turned into a P-DDHAP as $(aE, \{r_ibE\}_{i\in[m]}, \{r_icE\}_{i\in[m]})$ for $\{r_i\}_{i\in[m]} \xleftarrow{\$} G$.

In the following we will use this in the form $(aE, \{b_iE\}_{i\in[m]}, \{c_iE\}_{i\in[m]}) \approx_c (aE, \{c_ia^{-1}E\}_{i\in[m]}, \{c_iE\}_{i\in[m]})$.

2.3 Sigma protocol

A sigma protocol is a three message public coin proof of knowledge protocol. For a relation $R \subseteq X \times W$, where X is the space of statements and W is the space of witnesses, a sigma protocol for R consists of two proving algorithm P_1, P_2 and a verifying algorithm V. $P_1(x, w) \to (\mathsf{com}, st)$ outputs the first prover message com, named commitment, and a state st for P_2. The second message $\mathsf{ch} \xleftarrow{\$} \mathcal{C}$ from verifier, named challenge, honestly samples a challenge from the challenge space $\mathcal{C}$. Finally, $P_2(st, \mathsf{ch}) \to \mathsf{resp}$ outputs the third message resp, named response. The verifying algorithm $V(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) \to 0(\text{reject})/1(\text{accept})$ outputs whether the verifier accepts the transcript.

A sigma protocol should satisfy the following three properties.

Definition 4. *(Completeness) A sigma protocol is complete if for any $(x, w) \in R$, the probability*

$$\Pr\left[(\mathsf{com}, st) \leftarrow P_1(x, w), \mathsf{ch} \xleftarrow{\$} \mathcal{C}, \mathsf{resp} \leftarrow P_2(st, \mathsf{ch}), 0 \leftarrow V(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp})\right]$$

is negligible.

Definition 5. *(Honest Verifier Zero Knowledge/HVZK) Let* $\mathbf{Trans}(x, w) \to (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *be a function that honestly executes the sigma protocol and outputs* *a transcript. We say that the sigma protocol is HVZK if there exists a simulator* $\mathbf{Sim}(x) \to (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *such that the output distribution of* $\mathbf{Trans}(x, w)$ *and* $\mathbf{Sim}(x)$ *is indistinguishable.*

Definition 6. *(t-special soundness) A sigma protocol is t-special sound if there exist an efficient extractor* **Ext** *such that, for any set of t transcripts with the same* (x, com)*, denoted as* $(x, \mathsf{com}, \{\mathsf{ch}_i\}_{i\in[t]}, \{\mathsf{resp}_i\}_{i\in[t]})$*, where every* ch_i *is distinct, the probability*

$$\Pr\left[(x, s) \notin R \wedge \forall i \in [t], \mathsf{acc}_i = 1 : {\scriptstyle \begin{array}{c} \forall i\in[t],\ \mathsf{acc}_i \leftarrow V(x,\mathsf{com},\mathsf{ch}_i,\mathsf{resp}_i), \\ s\leftarrow \mathbf{Ext}(x,\mathsf{com},\{\mathsf{ch}_i\}_{i\in[t]},\{\mathsf{resp}_i\}_{i\in[t]}) \end{array}}\right]$$

is negligible.

Here, we formulate a more general form of special soundness. While most sigma protocol constructions in the literature adopt 2-special soundness, any t-special sound protocol with constant t can be similarly transformed into a signature scheme, simply by applying more rewinding trials.

2.4 The Forking Lemma

A sigma-protocol-based signature naturally allows witness extraction from the special soundness property. Through extracting the witness from signature forgeries, one can reduce the unforgeability property to the hardness of computing the witness. However, the main gap between special soundness and unforgeability is that special soundness needs multiple related transcripts to extract the witness, while a signature forging adversary only provides one. The forking lemma [30] is thus proposed to close this gap.

The concept of the forking lemma is as follows. In the random oracle model, let A be an adversary that can with non-negligible probability generate valid transcripts $(m, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ with $\mathsf{ch} = H(m, \mathsf{com})$. Since H is a random oracle, for some (m, com), A should be able to succeed on sufficiently many different ch' from H in order to achieve an overall non-negligible success probability. If we can rewind and rerun A with different oracle outputs on $H(m, \mathsf{com})$, we should be able to get multiple accepting transcripts.

To dig a little bit deeper, we can construct an efficient algorithm B that runs A as a subroutine, where $A \to (m, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ has at most Q oracle queries. The tuple (m, com) should, with all but negligible probability, be among one of the Q queries. B first guesses the *critical query* $i \in [Q]$, the index where $\mathcal{Q}_i = (m, \mathsf{com})$ is being queried. Then, B replays A with fixed random tape, fixed oracle outputs for the first $i - 1$ queries, and fresh random oracle outputs for the remaining queries. If the query guess i and fixed randomness are "good," which should happen with non-negligible probability, then among sufficiently many retries we should get t successful outputs of A, which are transcripts with identical (m, com) with distinct challenges ch's. For a rigorous proof, we refer the reader to [30, 31] for the forking lemma with 2 transcripts and [10] for a t-transcript version.

Here, we give a reformulated version of the improved forking lemma proposed by [10]. We renamed the variables to fit our notion and restrict parameters to the range that is sufficient for our proof.

Theorem 1. *(The Improved Forking Lemma [10], Reformulated) Let A be a probabilistic polynomial-time algorithm and Sim be a probabilistic polynomial-time simulator which can be queried by A. Let H be a random oracle with image size $|H| \geq 2^\lambda$. If A can output some valid tuple $(m, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ with non-negligible probability $\varepsilon \geq 1/poly(\lambda)$ within less than Q queries to the random oracle, then with $O(Qt \log t/\varepsilon)$ rewinds of A with different random oracles, A will, with at least constant probability, output t valid tuples $(m, \mathsf{com}, \mathsf{ch}_i, \mathsf{resp}_i)$ with identical (m, com) and pairwise distinct ch_i's.*

2.5 Group signature

A group signature scheme consists of one manager and n parties. The manager can set up a group and provide secret keys to each party. Every party is allowed to generate signatures on behalf of the whole group. Any party can verify the signature for the group without knowing the signer, while the manager party can open the signer's identity with his master secret key.

Syntax. A group signature scheme $\mathcal{GS}$ consists of the following four algorithms.

- **GKeygen**$(1^\lambda, 1^n) \to (\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk})$: The key generation algorithm **GKeygen** takes 1^λ and 1^n as inputs where λ is the security parameter and $n \in \mathbb{N}$ is the number of parties in the group, and outputs $(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk})$ where gpk is the public key for the group, sk_i being the secret key of the i-th player for each $i \in [n]$, and msk is the master secret key held by the manager for opening.
- **GSign**$(\mathsf{gpk}, m, \mathsf{sk}_k) \to \sigma$: The signing algorithm **GSign** takes a secret key sk_k and a message m as inputs, and outputs a signature σ of m using sk_k.
- **GVerify**$(\mathsf{gpk}, m, \sigma) \to y \in \{0, 1\}$: The verification algorithm **GVerify** takes the public key gpk, a message m, and a candidate signature σ as inputs, and outputs either 1 for accept or 0 for reject.
- **GOpen**$(\mathsf{gpk}, \mathsf{msk}, m, \sigma) \to k \in [n]$: The open algorithm **GOpen** takes the public key gpk, the manager's master secret key msk, a message m, and a signature σ as inputs, and outputs an identity k or abort with output $\perp$.

A group signature scheme should satisfy the following security properties.

Correctness. A group signature scheme is said to be correct if every honest signature can be correctly verified and opened.

Definition 7. *A group signature scheme $\mathcal{GS}$ is correct if for any tuple of keys* $(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathbf{GKeygen}(1^\kappa, 1^n)$*, any $i \in [n]$ and any message m,*

$$\Pr\left[\mathsf{acc}{=}1 \wedge \mathsf{out}{=}i : \begin{array}{c} \sigma\leftarrow\mathbf{GSign}(\mathsf{gpk},m,\mathsf{sk}_i), \\ \mathsf{acc}\leftarrow\mathbf{GVerify}(\mathsf{gpk},m,\sigma), \\ \mathsf{out}\leftarrow\mathbf{GOpen}(\mathsf{gpk},\mathsf{msk},m,\sigma) \end{array}\right] > 1 - \mathsf{negl}(\lambda)$$

Anonymity. A group signature is said to be anonymous if no adversary can determine the signer's identity among the group of signers given a signature, without using the master's secret key (msk).

Definition 8. *A group signature scheme $\mathcal{GS}$ is anonymous if for any PPT adversary A and any $n = poly(\lambda)$,*

$$\left|\Pr[1 \leftarrow G^{\mathsf{Anon}}_{A,0}(\lambda, n)] - \Pr[1 \leftarrow G^{\mathsf{Anon}}_{A,1}(\lambda, n)]\right| \leq \mathsf{negl}(\lambda),$$

where the game $G^{\mathsf{Anon}}_{A,b}(\lambda, n)$ is defined below.

$G^{\mathsf{Anon}}_{A,b}(\lambda, n)$: Anonymity game

```
1: $(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathbf{GKeygen}(1^\lambda, 1^n)$
2: $(st, i_0, i_1) \leftarrow A(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]})$
3: $b \leftarrow \{0, 1\}$
4: return $\mathsf{out} \leftarrow A^{\mathbf{GSign}(\mathsf{gpk},\cdot,\mathsf{sk}_{i_b})}(st)$
```

Unforgeability. A group signature is said to be unforgeable if no adversary can forge a valid signature that fails to open or opens to some non-corrupted parties, even if the manager has also colluded.

Definition 9. *A group signature scheme $\mathcal{GS}$ is unforgeable if for any PPT adversary A and any $n = poly(\lambda)$,*

$$\Pr[A \textit{ wins } G^{\mathsf{UF}}_{A}(\lambda, n)] < \mathsf{negl}(\lambda),$$

where the game $G^{\mathsf{UF}}_{A}(\lambda, n)$ is defined below.

$G^{\mathsf{UF}}_{A}(\lambda, n)$: Unforgeability game

```
1: $(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathbf{GKeygen}(1^\lambda, 1^n)$, $\mathsf{Cor} = \{\}$
2: $(m^*, \sigma^*) \leftarrow A^{\mathbf{GSign}(\mathsf{gpk},\cdot,\mathsf{sk}_i \notin \mathsf{Cor}),\mathbf{Corrupt}(\cdot)}(\mathsf{gpk}, \mathsf{msk})$
   {Corrupt($i$) returns $\mathsf{sk}_i$ stores query $i$ in list $\mathsf{Cor}$}
3: $A$ wins if $(m^*, \sigma^*)$ is not an output of GSign, $1 \leftarrow \mathbf{GVerify}(\mathsf{gpk}, m^*, \sigma^*)$
   and $i \leftarrow \mathbf{GOpen}(\mathsf{gpk}, \mathsf{msk}, m^*, \sigma^*)$ satisfies $i \notin \mathsf{Cor}$
```

2.6 Accountable ring signature

Accountable ring signatures (ARS) are a natural generalization for both group signatures and ring signatures. Compared to a group signature, ARS gives the power of group decision to the signer. On signing, the signer can sign for an arbitrary group (or ring, to fit the original naming), and can decide a master independent from the choice of the group. The master can open the identity of the signer among the group without needing to participate in the key generation of parties in the ring. Note that accountable ring signatures directly imply group signatures, simply by fixing the group and the master party at the key generation step. Thus, ARS can be viewed as a more flexible form of group signature.

Syntax. An accountable ring signature scheme $\mathcal{ARS}$ consists of the following algorithms.

- **MKeygen**$(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$ generates master public key/secret key pair.
- **Keygen**$(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ generates public key/secret key pair for group members.
- **Sign**$(\mathsf{mpk}, S = \{\mathsf{pk}_i\}_{i \in R}, m, \mathsf{sk}_{id}) \to \sigma$ generates a signature σ for message m for a set S with some secret key sk_{id} for $id \in R$.
- **Verify**$(\mathsf{mpk}, S = \{\mathsf{pk}_i\}_{i \in R}, m, \sigma) \to 1/0$ verifies whether (m, σ) is valid. **Verify** outputs 1 if verification passes and 0 otherwise.
- **Open**$(\mathsf{msk}, S = \{\mathsf{pk}_i\}_{i \in R}, m, \sigma) \to \mathsf{pk} \in S \cup \{\perp\}$ reveals the identity $\mathsf{pk} \in S$ for which the matching secret key was used to generate the signature σ. It outputs $id = \perp$ when the opening fails. (i.e. when σ is malformed)

We define the message space to be $\mathcal{M}$, the master public key space to be $\mathcal{K}_m$ and the public key space to be $\mathcal{K}$. We also define $\mathcal{KP}_m$ to be the set of all master key pairs $(\mathsf{mpk}, \mathsf{msk})$, and $\mathcal{KP}$ to be the set of all public/private key pairs $(\mathsf{pk}, \mathsf{sk})$. For simplicity, we keep the parameter λ implicit for the before-mentioned key spaces, and additionally require public keys to be all distinct for a set S of size $|S| \leq \mathsf{poly}(\lambda)$.

An accountable ring signature scheme should satisfy the following security properties.

Correctness. An ARS is said to be correct if every honest signature can be correctly verified and opened.

Definition 10. *An accountable ring signature scheme $\mathcal{ARS}$ is correct if for any master key pair $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{KP}_m$, any key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{KP}$, and any set of public keys S such that $\mathsf{pk} \in S$,*

$$\Pr\left[\mathsf{acc}=1 \wedge \mathsf{out}=\mathsf{pk} : \begin{array}{c} \sigma \leftarrow \mathbf{Sign}(\mathsf{mpk}, S, m, \mathsf{sk}), \\ \mathsf{acc} \leftarrow \mathbf{Verify}(\mathsf{mpk}, S, m, \sigma), \\ \mathsf{out} \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma) \end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Anonymity. An ARS is said to be anonymous if no adversary can determine the signer's identity within the set of signers of a signature without using the master secret key.

Definition 11. *An accountable ring signature scheme $\mathcal{ARS}$ is anonymous if for any PPT adversary A and any two key pairs $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{KP}$,*

$$\left| \begin{array}{l} \Pr_{(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathbf{MKeygen}(1^\lambda)} [1 \leftarrow A^{\mathbf{Sign}^*(\mathsf{mpk},\cdot,\cdot,\mathsf{sk}_0)}(\mathsf{mpk})] \\ - \Pr_{(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathbf{MKeygen}(1^\lambda)} [1 \leftarrow A^{\mathbf{Sign}^*(\mathsf{mpk},\cdot,\cdot,\mathsf{sk}_1)}(\mathsf{mpk})] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Where $\mathbf{Sign}^*(\mathsf{mpk}, S, m, \mathsf{sk}_b)$ is an oracle that returns an honest signature only when both $\mathsf{pk}_0, \mathsf{pk}_1 \in S$. Otherwise, $\mathbf{Sign}^*$ aborts.

Unforgeability. An ARS is said to be unforgeable if no adversary can forge a valid signature that fails to open or opens to some non-corrupted party, even if the manager has also colluded.

We model this property with the unforgeability game $G^{\mathsf{UF}}_{n_h}$. Among n_h honest keys pairs, the adversary A can call the signing oracle to obtain honest signatures, or call the corruption oracle to obtain the secret keys of the honest parties. The adversary wins if it outputs a valid signature that opens to a non-corrupted party or fails to open. We abuse the notation $sk_i \in \mathsf{Hon}$ if $pk_i \in \mathsf{Hon}$.

G^{UF}_{A,n_h}: Unforgeability game

```
1: $\forall i \in [n_h], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathbf{Keygen}(1^\lambda)$. Let $\mathsf{Hon} = \{\mathsf{pk}_i\}_{i \in [n_h]}, \mathsf{Cor} = \{\}$.
2: $(S, m^*, \sigma^*) \leftarrow A^{\mathbf{Sign}(\cdot,\cdot,\cdot,\mathsf{sk}_i \in \mathsf{Hon}), \mathbf{Corrupt}(\cdot)}(\mathsf{Hon})$
   $\{\mathbf{Corrupt}(\mathsf{pk}_i)$ returns $\mathsf{sk}_i$ for $\mathsf{pk}_i \in \mathsf{Hon}$ and stores query $\mathsf{pk}_i$ in list $\mathsf{Cor}\}$
3: $A$ wins if $(m^*, \sigma^*)$ is not an output of $\mathbf{Sign}$, $1 \leftarrow \mathbf{Verify}(\mathsf{mpk}, S, m^*, \sigma^*)$ and
   $\mathsf{pk} \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma^*)$ satisfies $\mathsf{pk} \in \{\perp\} \cup \mathsf{Hon} \setminus \mathsf{Cor}$
```

Definition 12. *An accountable ring signature scheme $\mathcal{ARS}$ is unforgeable if for any PPT adversary A, any valid master key pair $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{KP}_m$ and any $n_h = poly(\lambda)$*

$$\Pr[A \textit{ wins } G^{\mathsf{UF}}_{A,n_h}(\mathsf{mpk}, \mathsf{msk})] < \mathsf{negl}(\lambda).$$

Transforming ARS to GS. As mentioned earlier, an accountable ring signature can be viewed as a generalization of a group signature. We give here the general transformation from an ARS scheme $\mathcal{ARS}$ to a group signature scheme $\mathcal{GS}^{\mathcal{ARS}}$.

The algorithms of the group signature scheme $\mathcal{GS}^{\mathcal{ARS}}$ are detailed as follows:

– $\mathbf{GKeygen}(1^\lambda, 1^n)$:

```
1: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{ARS}.\mathbf{MKeygen}(1^\lambda)$
2: $\forall i \in [n], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathcal{ARS}.\mathbf{Keygen}(1^\lambda)$, Let $S = \{\mathsf{pk}_i\}_{i \in [n]}$ and $\mathsf{gpk} = (\mathsf{mpk}, S)$
```

```
    3: return $(\mathsf{gpk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk})$
– GSign($\mathsf{gpk} = (\mathsf{mpk}, S), m, \mathsf{sk}_k$)
    1: return $\sigma \leftarrow \mathcal{ARS}.\mathbf{Sign}(\mathsf{mpk}, S, m, \mathsf{sk}_k)$
– GVerify($\mathsf{gpk} = (\mathsf{mpk}, S), m, \sigma$):
    1: return $\sigma \leftarrow \mathcal{ARS}.\mathbf{Verify}(\mathsf{mpk}, S, m, \sigma)$
– GOpen($\mathsf{gpk} = (\mathsf{mpk}, S), \mathsf{msk}, m, \sigma$):
    1: $\mathsf{pk} \leftarrow \mathcal{ARS}.\mathbf{Open}(\mathsf{msk}, S, m, \sigma)$
    2: return $k$ s.t. $\mathsf{pk} = \mathsf{pk}_k \in S$ or $\perp$ otherwise
```

Note that the transformation only changes the formulation of the setup stage. Thus, the security properties from $\mathcal{ARS}$ transfer directly to the induced group signature scheme $\mathcal{GS}^{\mathcal{ARS}}$.

3 Openable OR-Sigma Protocol

In this section, we will introduce the openable sigma protocol, which is an intermediate primitive toward group signatures and accountable ring signatures. We will first give some intuition on how we formulate this primitive, and then give a formal definition and construction from DDH-hard group actions.

3.1 Intuition

A typical construction of a Fiat-Shamir based signature starts from a sigma protocol. As introduced in Section 2.3, the three message protocol $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ only requires special soundness, which is, informally speaking, weaker than the unforgeability property in the sense that multiple transcripts are required in order to break the underlying hardness. The forking lemma closes this gap with the power of rewinding and random oracle programming. As stated in Section 2.4, the lemma takes a forger that outputs a single forgery and gives an algorithm that outputs multiple instances of valid $(\mathsf{com}, \mathsf{ch}_j, \mathsf{resp}_j)$'s. This gives a transformation from a signature breaker to a witness extractor, bridging the two security notions.

For our accountable ring signature, we thus plan to follow the previous roadmap. We design a sigma protocol that supports an extra "opening" property. Our openable sigma protocol takes n statements as input, and additionally requires the prover to take a master public key mpk as input on generating the first message com. The function **Open**, with the master secret key msk, can then extract the actual statement to which the proving witness corresponds to. For a com generated from statement $(x_1, \ldots, x_n)$ and witness w_i with $(x_i, w_i) \in R$, we have $x_i = \mathbf{Open}(\mathsf{com}, \mathsf{msk})$. As our target is a signature scheme, (x_i, w_i) would be set to public key/secret key pairs, and thus the open function outputs the signer's identity.

To achieve the stronger security property of ARS after the Fiat-Shamir transformation, our openable sigma protocol needs to have modified security properties correspondingly. For *special soundness*, we would not be satisfied with extracting only "one of the witnesses," instead we need to build an extractor that extracts a witness which matches the opening result. Such a stronger extractor will allow us to extract secret keys from adversaries that can impersonate other players. For *honest verifier zero knowledge* (HVZK), we require the transcript to be ZK even when given the master secret key msk. This is crucial for proving that the impersonating attack cannot succeed even with a corrupted manager. Note that when given msk, one cannot hope to hide the signer's identity, so we only require ZK against the signer's witness. The formulation for the HVZK simulator thus takes the signer identity as input. Finally, we need an extra property to provide anonymity for the signer, which we named *computational witness indistinguishability* (CWI). CWI requires that, given honest master key pairs, the transcript generated from two different witnesses/identities should be indistinguishable. This property is formulated as the indistinguishability of two signing oracles.

3.2 Definition

An openable sigma protocol Σ_λ with security parameter λ is defined with respect to two relations. A base relation $(x, s) \in R^\lambda \subset X \times W$, and an efficiently samplable opening relation $\mathbf{MKeygen}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk}) \in R_m^\lambda$. We will omit the superscripts λ when there is no ambiguity. We also define the or-relation for R. $(\{x_i\}_{i\in[n]}, s) \in R_n$ if and only if all x_i are distinct and $\exists i \in [n]$ s.t. $(x_i, s) \in R$

The openable sigma protocol Σ contains the following four algorithms.

– $\mathbf{Commit}(x_m, \{x_i\}_{i\in[n]}, s) \to (\mathsf{com}, st)$ generates a commitment com based on $(\{x_i\}_{i\in[n]}, s) \in R_n$. **Commit** also generates a state st which is shared with **Resp** and will be kept implicit for convenience.
– $\mathbf{Resp}(x_m, \{x_i\}_{i\in[n]}, s, \mathsf{com}, \mathsf{ch}, st) \to \mathsf{resp}$ computes a response resp relative to a challenge $\mathsf{ch} \xleftarrow{\$} \mathcal{C}$.
– $\mathbf{Verify}(x_m, \{x_i\}_{i\in[n]}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) \to 1/0$ verifies whether a tuple $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ is valid. **Verify** outputs 1 if the verification passes and 0 otherwise.
– $\mathbf{Open}(s_m, \{x_i\}_{i\in[n]}, \mathsf{com}) \to x \in \{x_i\}_{i\in[n]} \cup \{\perp\}$ reveals some $(x, s) \in R$, where s is the witness used to generate the commitment com. It outputs $x =\perp$ when the opening fails. (i.e. when com is malformed)

An openable OR sigma protocol should satisfy the following properties.

Definition 13. *(Completeness): An openable sigma protocol Σ is complete if for all $n = poly(\lambda)$, $(x_m, s_m) \in R_m$, $(\{x_i\}_{i\in[n]}, s) \in R_n$, $\mathsf{ch} \in \mathcal{C}$, and $x \in \{x_i\}_{i\in[n]}$*

such that $(x, s) \in R$,

$$\Pr\left[\mathsf{acc} = 1 \wedge id = x : \begin{smallmatrix} \mathsf{com}\leftarrow\mathbf{Commit}(x_m,\{x_i\}_{i\in[n]},s), \\ \mathsf{resp}\leftarrow\mathbf{Resp}(x_m,\{x_i\}_{i\in[n]},s,\mathsf{com},\mathsf{ch}), \\ \mathsf{acc}\leftarrow\mathbf{Verify}(\{x_m,\{x_i\}_{i\in[n]},\mathsf{com},\mathsf{ch},\mathsf{resp}), \\ id\leftarrow\mathbf{Open}(s_m,\{x_i\}_{i\in[n]},\mathsf{com}) \end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 14. *(t-Special Soundness):An openable sigma protocol Σ is t-special sound if for all $n = poly(\lambda)$ there exists an efficient extractor* **Ext** *such that, for all $(x_m, s_m) \in R_m$ and any $(\{x_i\}_{i\in[n]}, \mathsf{com}, \{\mathsf{ch}_j\}_{j\in[t]}, \{\mathsf{resp}_j\}_{j\in[t]})$ such that each $\mathsf{ch}_j \in \mathcal{C}$ are distinct, then*

$$\Pr\left[\begin{smallmatrix}(\forall j\in[t],\ \mathsf{acc}_j=1)\wedge \\ (x=\perp\vee(x,s)\notin R)\end{smallmatrix} : \begin{smallmatrix} \forall j\in\mathcal{C},\ \mathsf{acc}_j\leftarrow\mathbf{Ver}(x_m,\{x_i\}_{i\in[n]},\mathsf{com},\mathsf{ch}_j,\mathsf{resp}_j), \\ x\leftarrow\mathbf{Open}(s_m,\{x_i\}_{i\in[n]},\mathsf{com}), \\ s\leftarrow\mathbf{Ext}(\{x_i\}_{i\in[n]},\mathsf{com},\{\mathsf{ch}_j\}_{j\in[t]},\{\mathsf{resp}_j\}_{j\in[t]}) \end{smallmatrix}\right] = 0.$$

Definition 15. *(Statistical Honest Verifier Zero Knowledge / sHVZK):An openable sigma protocol Σ is statistical HVZK if there exists an efficient simulator* **Sim** *such that, for any $x_m \in X_m$, any $(\{x_i\}_{i\in[n]}, s) \in R_n$, and $x \in \{x_i\}_{i\in[n]}$ such that $(x, s) \in R$,*

$$\mathbf{Trans}(x_m, \{x_i\}_{i\in[n]}, s) \approx_s \mathbf{Sim}(x_m, \{x_i\}_{i\in[n]}, x)$$

where **Trans** *outputs honest transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *generated honestly by* **Commit** *and* **Resp** *with honestly sampled* $\mathsf{ch} \xleftarrow{\$} \mathcal{C}$.

Definition 16. *(Computational Witness Indistinguishability / CWI):An openable sigma protocol Σ is computational witness indistinguishable with respect to an efficient instance generator* $(\mathsf{mpk}, \mathsf{msk}) \in R_m \leftarrow \mathbf{MKeygen}(1^\lambda)$ *if, for any two $(x_i, s_i), (x_j, s_j) \in R$ and any efficient adversary Adv,*

$$\left| \begin{array}{l} \Pr\limits_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}}[1 \leftarrow Adv^{\mathbf{Trans}^*(\mathsf{mpk},\cdot,s_i)}(\mathsf{mpk})] \\ - \Pr\limits_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}}[1 \leftarrow Adv^{\mathbf{Trans}^*(\mathsf{mpk},\cdot,s_j)}(\mathsf{mpk})] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where $\mathbf{Trans}^*(\mathsf{mpk}, S, \cdot)$ *returns an honest transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *tuple from Σ if both $x_i, x_j \in S$ and aborts otherwise.*

3.3 Construction

Here, we give our construction to an openable or sigma protocol $\Sigma_{GA,\lambda}$ for relations from our DDH-secure group action $\mathcal{GA}_\lambda = (G, \mathcal{E})$. We let $E \in \mathcal{E}$ be some fixed element in $\mathcal{E}$. When implemented with CSIDH, we can choose the curve $E_0 : y^2 = x^3 + x$ for simplicity. Let the relation $R_E = \{(aE, a) | a \in G\} \subset \mathcal{E} \times G$.

For our Σ_{GA}, we define its opening relation $R_m = R_E$, with the natural instance generator $\mathbf{MKeygen}(1^\lambda)$ that samples $a \xleftarrow{\$} G$ and outputs (aE, a).

The base relation is also set to R_E. For inputs $E_m \in \mathcal{E}$ and $(\{E_i\}_{i\in[n]}, s) \in R_n$ with any $n = poly(\lambda)$, the algorithms for Σ_{GA} are constructed as follow.

– **Commit**$(E_m, \{E_i\}_{i\in[n]}, s)$

```
1: set $k \in [n]$ s.t. $(E_k, s) \in R$.
2: $\{\Delta_i\}_{i\in[n]}, \{\Delta'_i\}_{i\in[n]}, b \xleftarrow{\$} G$
3: $\tau \xleftarrow{\$} sym(n)$ {$\tau$ is a random permutation}
4: $\forall i \in [n] : E_i^\alpha := \Delta_i E_i$
5: $\forall i \in [n] : E_i^\beta := \Delta'_i E_i^\alpha = \Delta_i \Delta'_i E_i$
6: $\forall i \in [n] : E_i^\gamma := b E_i^\beta = \Delta_i \Delta'_i b E_i$
7: $E^{\mathsf{Open}} := \Delta_k \Delta'_k s E_m$
8: $E^{\mathsf{Check}} := \Delta_k \Delta'_k b s E_m = b E^{\mathsf{Open}}$
9: $\mathsf{st} = (\{\Delta_i\}_{i\in[n]}, \{\Delta'_i\}_{i\in[n]}, b, l = \Delta_k \Delta'_k b s)$
10: **return** $(\mathsf{com}, st) = ((\{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in[n]}, \tau(\{E_i^\gamma\}_{i\in[n]}), E^{\mathsf{Open}}, E^{\mathsf{Check}}), st)$
    {We use $\tau(\cdot)$ as a lazy convention of sending a permuted list}
```

– **Resp**$(E_m, \{E_i\}_{i\in[n]}, s, \mathsf{com}, \mathsf{ch}, st)$:

```
1: **if** $\mathsf{ch} = 1$ **then**
2:     **return** $\mathsf{resp} := \{\Delta_i\}_{i\in[n]}$
3: **if** $\mathsf{ch} = 2$ **then**
4:     **return** $\mathsf{resp} := \{\Delta'_i\}_{i\in[n]}$
5: **if** $\mathsf{ch} = 3$ **then**
6:     **return** $\mathsf{resp} := b$
7: **if** $\mathsf{ch} = 4$ **then**
8:     **return** $\mathsf{resp} := l = \Delta_k \Delta'_k b s$
```

– **Verify**$(E_m, \{E_i\}_{i\in[n]}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$:

```
1: **return** 0 if $\{E_i\}_{i\in[n]}$ or $\{E_i^\beta\}_{i\in[n]}$ are not all distinct
2: **if** $\mathsf{ch} = 1$ **then**
3:     **check** $\forall i \in [n] : E_i^\alpha = \Delta_i E_i$
4: **if** $\mathsf{ch} = 2$ **then**
5:     **check** $\forall i \in [n] : \Delta'_i E_i^\alpha = E_i^\beta$
6: **if** $\mathsf{ch} = 3$ **then**
7:     **check** $\exists \tau' \in sym(n)$ s.t. $\tau'(\{b E_i^\beta\}_{i\in[n]}) = \tau(\{E_i^\gamma\}_{i\in[n]})$
8:     **check** $E^{\mathsf{Check}} = b E^{\mathsf{Open}}$
9: **if** $\mathsf{ch} = 4$ **then**
10:    **check** $E^{\mathsf{Check}} = l E_m$
11:    **check** $\exists E^\gamma \in \tau(\{E_i^\gamma\}_{i\in[n]})$ $s.t.$ $E^\gamma = lE$
12: **return** 1 **if all checks pass**
```

– **Open**$(s_m := \mathsf{msk}, \{E_i\}_{i\in[n]} := \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{com})$:

```
1: **for** $i \in [n]$ **do**
2:     **if** $s_m E_i^\beta = E^{\mathsf{Open}}$ **then**
3:         **return** $E_i$
4: **return** $\perp$
```

The construction of our openable sigma protocol looks complicated, but the intuition is simple. The core section of the message com is $(E^{\beta}, E^{\mathsf{Open}})$, which allows opening. The other parts of com are to ensure that the opening section is honestly generated. E^{α} along with the challenge/response pair on $\mathsf{ch} = 1, 2$ allows extraction for $\Delta_i \Delta_i'$'s, ensuring that E^{β} is honestly generated. $(E^{\gamma}, E^{\mathsf{Check}})$ along with the challenge/response pair on $\mathsf{ch} = 3, 4$ verifies the relation between E^{β} and E^{Open}. By using a permuted E^{γ}, the CWI property is preserved through such a verification process. Combined together, we complete the proof of knowledge protocol.

Theorem 2. *Σ_{GA} is an openable sigma protocol with R_E being both the opening relation and the base relation*

3.4 Security

The proof for Theorem 2 can be broken down into proving each of the required properties.

Lemma 1. *Σ_{GA} is **complete***

Proof. By the definition of **Commit** and **Verify**, any honestly generated $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ based on $(\{E_i\}_{i\in[n]}, s) \in R_n$ will be accepted as long as the set $\{E_i^{\beta}\}_{i\in[n]}$ is pairwise distinct. Since $\mathcal{GA}$ is free and transitive, there is a unique $g \in G$ s.t. $gE_i = E_j$. Thus, $E_i^{\beta} = E_j^{\beta}$ if and only if $(\Delta_j \Delta_j')^{-1} \Delta_i \Delta_i' = g$, which happens with negligible probability since all Δ's are honestly sampled. Hence with probability $1 - n \cdot \mathsf{negl}(\lambda)$, the set $\{E_i^{\beta}\}_{i\in[n]}$ are all distinct, and hence **Verify** accepts.

For the function **Open**, note that if $(E_m, s_m) \in R_m$ and $(E_k, s) \in R$, then $E^{\mathsf{Open}} = \Delta_k \Delta_k' s E_m = \Delta_k \Delta_k' s s_m E$, hence $s_m E_k^{\beta} = E^{\mathsf{Open}}$. As argued previously, $\{E_i^{\beta}\}_{i\in[n]}$ are all distinct with probability $1 - \mathsf{negl}(\lambda)$, and k would be unique if this is the case. Thus the probability that **Open** outputs E_k is overwhelming, concluding the proof that Σ_{GA} is complete.

□

Lemma 2. *Σ_{GA} is **4-special sound***

Proof. For any $E_m \in \mathcal{E}$ and any $(\{E_i\}_{i\in[n]}, \mathsf{com}, \{\mathsf{resp}_j\}_{j\in\mathcal{C}})$ where $\mathsf{com} = (\{E_i^{\alpha}\}_{i\in[n]}, \{E_i^{\beta}\}_{i\in[n]}, \sigma(\{E_i^{\gamma}\}_{i\in[n]})$, and $\{\mathsf{resp}_j\}_{j\in[4]} = (\{\Delta_i\}_{i\in[n]}, \{\Delta_i'\}_{i\in[n]}, b, l)$. Suppose that $\forall j \in [4], 1 \leftarrow \mathbf{Ver}(E_m, \{E_i\}_{i\in[n]}, \mathsf{com}, j, \mathsf{resp}_j)$, then by the definition of **Verify**, we can get the following equations:

$$\begin{cases} \{E_i\}_{i\in[n]}, \{E_i^\beta\}_{i\in[n]} \text{ are both pairwise distinct sets} \\ \forall i \in [n] : E_i^\alpha = \Delta_i E_i, E_i^\beta = \Delta_i' E_i^\alpha \\ \exists \tau' \in sym(n) \; s.t. \; \tau'(\{bE_i^\beta\}_{i\in[n]}) = \tau(\{E_i^\gamma\}_{i\in[n]}) \\ \exists E^\gamma \in \tau(\{E_i^\gamma\}_{i\in[n]}) \; s.t. \; E^\gamma = lE \\ E^{\mathsf{Check}} = lE_m = bE^{\mathsf{Open}} \end{cases}$$

Thus, there exists a unique $k \in [n]$ such that $lE = bE_k^\beta = \Delta_k' bE_k^\alpha = \Delta_k \Delta_k' bE_k$, which means $l(\Delta_k \Delta_k' b)^{-1} E = E_k$. This implies that $(E_k, l(\Delta_k \Delta_k' b)^{-1}) \in R_E$. Furthermore, we also have $E^{\mathsf{Open}} = b^{-1} lE_m = s_m b^{-1} lE = s_m E_k^\beta$. This implies that $E_k \leftarrow \mathbf{Open}(s_m, \{E_i\}_{i\in[n]}, \mathsf{com})$. Thus **Open** does not output $\perp$. From these observations, we can easily construct the extractor $\mathbf{Ext}(\mathsf{com}, \{\mathsf{resp}_j\}_{j\in\mathcal{C}})$, which simply searches through $k \in [n]$ for k satisfying $lE = bE_k^\beta$, then output $s = l(\Delta_k \Delta_k' b)^{-1}$. This concludes the proof that Σ_{GA} is **4-special sound**. □

Lemma 3. *Σ_{GA} is* ***statistically honest verifier zero-knowledge****.*

Proof. The construction of **Sim** is given in the following algorithm. We will show that **Sim** is in fact a perfect simulator for **Trans**.

```
Sim($E_m, \{E_i\}_{i\in[n]}, E_k$)
1:  $\mathsf{ch} \xleftarrow{\$} \{1,2,3,4\}$
2:  $b \xleftarrow{\$} G$, $\tau \xleftarrow{\$} sym(n)$
3:  if $\mathsf{ch} = 1$ then
4:      $\{\Delta_i\}_{i\in[n]}, \{D_i\}_{i\in[n]} \xleftarrow{\$} G$
5:      $\forall i \in [n] : E_i^\alpha := \Delta_i E_i$
6:      $\forall i \in [n] : E_i^\beta := \Delta_i D_i E$
7:      $E^{\mathsf{Open}} := \Delta_k D_k E_m$
8:  else if $\mathsf{ch} = 2,3,4$ then
9:      $\{D_i\}_{i\in[n]}, \{\Delta_i'\}_{i\in[n]} \xleftarrow{\$} G$
10:     $\forall i \in [n] : E_i^\alpha := D_i E$
11:     $\forall i \in [n] : E_i^\beta := \Delta_i' E_i^\alpha$
12:     $E^{\mathsf{Open}} := D_k \Delta_k' E_m$
13: $\forall i \in [n] : E_i^\gamma := bE_i^\beta$
14: if $\mathsf{ch} = 1,2,3$ then
15:     $E^{\mathsf{Check}} := bE^{\mathsf{Open}}$
16: else if $\mathsf{ch} = 4$ then
17:     $l := \Delta_k D_k b$
18:     $E_{km}^{\mathsf{Check}} := lE_m$
19: return $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ with the same definition as honest Commit and Resp
```

Since $\mathcal{GA}$ is free and transitive, for every $E_i \in \mathcal{E}$, there exists a unique $s_i \in G$ s.t. $s_i E = E_i$. In Sim_1, we can thus set $\Delta_i' = D_i s_i^{-1}$ in case $\mathsf{ch} = 1$ and $\Delta_i = D_i s_i^{-1}$ in case $\mathsf{ch} = 2,3,4$. Since the distribution of $D_i s_i^{-1}$ is uniformly random, **Sim** generates identical distributions for Δ's as **Trans**. Thus the output distribution of **Sim** should also be identical to the real transcript. Checking that verfication passes for all cases shows that **Sim** is a perfect simulator. □

Lemma 4. *Σ_{GA} is computational witness indistinguishable with respect to the natural instance generator* **MKeygen***, (assuming DDHAP is hard for $\mathcal{GA}$).*

Here we will finally use the fact that $\mathcal{GA}$ is DDH-hard. We will prove this theorem through two hybrids. We highlight the changes between **Trans** and Hyb_1 and between Hyb_1 and Hyb_2 with different colors for easier comparison.

Lemma 5. *For any $(\{x_i\}_{i\in[n]}, s) \in R_n$ and for any efficient adversary Adv*

$$\left| \begin{array}{l} \Pr_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}(1^\lambda)}[1 \leftarrow Adv^{\mathbf{Trans}(\mathsf{mpk},\{x_i\}_{i\in[n]},s)}(\mathsf{mpk})] \\ -\Pr_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}(1^\lambda)}[1 \leftarrow Adv^{\mathbf{Hyb_1}(\mathsf{mpk},\{x_i\}_{i\in[n]},s)}(\mathsf{mpk})] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

$\mathbf{Hyb}_1(E_m, \{E_i\}_{i\in[n]}, s)$

```
1: $\mathsf{ch} \xleftarrow{\$} \{1,2,3,4\}$
2: set $k \in [n]$ s.t. $(E_k, s) \in R$.
3: $\{\Delta_i\}_{i\in[n]}, \{\Delta'_i\}_{i\in[n]}, b \xleftarrow{\$} G$
4: $\tau \xleftarrow{\$} sym(n)$
5: $\forall i \in [n] : E_i^\alpha = \Delta_i E_i,\ E_i^\beta = \Delta'_i E_i^\alpha$
6: $\forall i \in [n] : E_i^\gamma = bE_i^\beta$
7: $r \xleftarrow{\$} G,\ E^{\mathsf{Open}} = rE$
8: if $\mathsf{ch} = 1,2,3$ then
9:     $E^{\mathsf{Check}} = bE^{\mathsf{Open}}$
10: else if $\mathsf{ch} = 4$ then
11:     $l = \Delta_k\Delta'_k bs,\ E^{\mathsf{Check}} = lE_m$
12: set $\mathsf{resp}$ honestly w.r.t $\mathsf{ch}$
13: return $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

Proof. We first note that the difference between honest transcript **Trans** and $\mathbf{Hyb}_1$ is that $\mathbf{Hyb}_1$ replaces honest E^{Open} with rE for a random $r \in G$. For $\mathsf{ch} \neq 4$, E^{Check} is also replaced accordingly to E^{Open}.

We will prove the indistinguishability of $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow$ **TRANS** and $(\mathsf{com}', \mathsf{ch}', \mathsf{resp}') \leftarrow \mathbf{Hyb}_1$ for each different challenge $\mathsf{ch} \in \mathcal{C}$ separately. In the following proof, we set k s.t. $(E_k, s) \in R$, as in both **TRANS** and $\mathbf{Hyb}_1$

For $\mathsf{ch}' = 1$, we have $\mathsf{resp}' = \{\Delta_i\}_{i\in[n]}$, which is honestly generated and thus identical to **Trans**. We thus focus on the com' part.

By the hardness of P-DDHAP, for random $\Delta'_k, r \xleftarrow{\$} G$, we have

$$(E_m, \Delta'_k E, \Delta'_k E_m) \approx_c (E_m, \Delta'_k E, rE)$$

Hence, for random $\Delta_k, \Delta'_k, b, r \xleftarrow{\$} G$ and honestly generated $(E_m, E_k^\beta, E_k^\gamma, E^{\mathsf{Open}}, E^{\mathsf{Check}})$, we have

$$\begin{aligned} &(E_m, E_k^\beta, E_k^\gamma, E^{\mathsf{Open}}, E^{\mathsf{Check}}) \\ =&(E_m, \Delta_k s(\Delta'_k E), \Delta_k bs(\Delta'_k E), \Delta_k s(\Delta'_k E_m), \Delta_k bs(\Delta'_k E_m)) \\ \approx_c&(E_m, \Delta_k s(\Delta'_k E), \Delta_k bs(\Delta'_k E), \Delta_k s(rE), \Delta_k bs(rE)) \\ =&(E_m, E_k^\beta, E_k^\gamma, r'E, br'E) \end{aligned}$$

Where LHS is the output com from **Trans**, restricted to the variables dependent on s_m or Δ'_k. RHS is the corresponding partial output from $\mathbf{Hyb}_1$. As the remaining parts of **Trans** and $\mathbf{Hyb}_1$ are equivalent, this equation shows that the output distributions of **Trans** and $\mathbf{Hyb}_1$ are indistinguishable for $\mathsf{ch} = 1$.

For the case $\mathsf{ch} = 2, 3$, the indistinguishability can be proved in a similar fashion. Notice again that for random $\Delta_k, r \xleftarrow{\$} G$, $(E_m, \Delta_k E, \Delta_k E_m) \approx_c (E_m, \Delta_k E, rE)$. Thus for random $\Delta_k, \Delta'_k, b, r \xleftarrow{\$} G$

$$\begin{aligned}
&(E_m, E_k^\alpha, E_k^\beta, E_k^\gamma, E^{\mathsf{Open}}, E^{\mathsf{Check}})\\
=&(E_m, s(\Delta_k E), \Delta'_k s(\Delta_k E), \Delta'_k bs(\Delta_k E), \Delta'_k s(\Delta_k E_m), \Delta'_k bs(\Delta_k E_m))\\
\approx_c&(E_m, s(\Delta_k E), \Delta'_k s(\Delta_k E), \Delta'_k bs(\Delta_k E), \Delta'_k s(rE), \Delta'_k bs(rE))\\
=&(E_m, E_k^\alpha, E_k^\beta, E_k^\gamma, r'E, br'E)
\end{aligned}$$

For the case $\mathsf{ch} = 4$, we would need a slight change. First we recall the fact that, since $\mathcal{GA}$ is free and transitive, for every E_i there exists a unique $s_i \in G$ s.t. $s_i E = E_i$. Thus, sampling $\{D_i\}_{i\in[n]}, b \xleftarrow{\$} G$ and letting $\Delta'_i = (bs_i)^{-1} D_i$ gives us a uniformly distributed $\{\Delta'_i\}_{i\in[n]}$.

Now, again from P-DDHAP, for random $b, r \xleftarrow{\$} G$,

$$(E_m, b^{-1}E, b^{-1}E_m) \approx_c (E_m, b^{-1}E, rE)$$

Thus, for random $\{\Delta_i\}_{i\in[n]}, \{D_i\}_{i\in[n]}, b, r \xleftarrow{\$} G$ where $D_i = \Delta'_i bs_i$, we have

$$\begin{aligned}
&(E_m, \{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in[n]}, \{E_i^\gamma\}_{i\in[n]}, E^{\mathsf{Open}}, E^{\mathsf{Check}}, l)\\
=&(E_m, \{\Delta_i E_i\}_{i\in[n]}, \{\Delta_i \Delta'_i E_i\}_{i\in[n]}, \{\Delta_i \Delta'_i bE_i\}_{i\in[n]}, \Delta_k \Delta'_k s_k E_m, \Delta_k \Delta'_k bs_k E_m, \Delta_k \Delta'_k bs_k)\\
=&(E_m, \{\Delta_i E_i\}_{i\in[n]}, \{\Delta_i D_i(b^{-1}E)\}_{i\in[n]}, \{\Delta_i D_i E\}_{i\in[n]}, \Delta_k D_k(b^{-1}E_m), \Delta_k D_k E_m, \Delta_k D_k)\\
\approx_c&(E_m, \{\Delta_i E_i\}_{i\in[n]}, \{\Delta_i D_i(b^{-1}E)\}_{i\in[n]}, \{\Delta_i D_i E\}_{i\in[n]}, \Delta_k D_k(rE), \Delta_k D_k E_m, \Delta_k D_k)\\
=&(E_m, \{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in[n]}, \{E_i^\gamma\}_{i\in[n]}, r'E, E^{\mathsf{Check}}, l)
\end{aligned}$$

Finally, since both ch and ch' are sampled randomly in $\{1, 2, 3, 4\}$, we can conclude that **TRANS** and $\mathbf{Hyb}_1$ are computationally indistinguishable. □

Lemma 6. *For any* $(\{x_i\}_{i\in[n]}, s) \in R_n$,

$$\left|\begin{array}{l} \Pr_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}(1^\lambda)}[1 \leftarrow Adv^{\mathbf{Hyb_1}(\mathsf{mpk},\{x_i\}_{i\in[n]},s)}(\mathsf{mpk})] \\ -\Pr_{(\mathsf{mpk},\mathsf{msk})\leftarrow\mathbf{MKeygen}(1^\lambda)}[1 \leftarrow Adv^{\mathbf{Hyb_2}(\mathsf{mpk},\{x_i\}_{i\in[n]},s)}(\mathsf{mpk})] \end{array}\right| \le \mathsf{negl}(\lambda).$$

$\mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s)$

```
1: $\mathsf{ch} \xleftarrow{\$} \{1,2,3,4\}$
2: set $k \in [n]$ s.t. $(E_k, s) \in R$.
3: $\{\Delta_i\}_{i\in[n]}, \{\Delta'_i\}_{i\in[n]}, b \xleftarrow{\$} G$
4: $\tau \xleftarrow{\$} sym(n)$
5: $\forall i \in [n] : E_i^\alpha = \Delta_i E_i,\ E_i^\beta = \Delta'_i E_i^\alpha$
6: $r \xleftarrow{\$} G,\ E^{\mathsf{Open}} = rE$
7: if $\mathsf{ch} = 1,2,3$ then
8:     $E^{\mathsf{Check}} = bE^{\mathsf{Open}}$
9:     $\forall i \in [n] : E_i^\gamma = bE_i^\beta$
10: else if $\mathsf{ch} = 4$ then
11:     $\forall i \in [n] : r_i \xleftarrow{\$} G, E_i^\gamma = r_i E$
12:     $l = r_k,\ E^{\mathsf{Check}} = lE_m$
13: set $\mathsf{resp}$ honestly w.r.t $\mathsf{ch}$
14: return $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

Proof. The hybrids $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ differ only in the case $\mathsf{ch} = 4$, in which we replace the whole E^γ with random curves, E^{Check} and l are also changed correspondingly. As in the previous proof, we use the fact that sampling $\{D_i\}_{i\in[n]}, b \xleftarrow{\$} G$ and letting $\Delta'_i = (bs_i)^{-1}D_i$ gives us uniformly random $(\{\Delta'_i\}_{i\in[n]}, b)$

By P-DDHAP, for random $b, \{D_i\}_{i\in[n]\setminus\{k\}}, \{r_i\}_{i\in[n]\setminus\{k\}}$,

$$\begin{aligned}&(b^{-1}E, \{D_iE\}_{i\in[n]\setminus\{k\}}, \{D_ib^{-1}E\}_{i\in[n]\setminus\{k\}})\\ \approx_c&(b^{-1}E, \{r_iE\}_{i\in[n]\setminus\{k\}}, \{D_ib^{-1}E\}_{i\in[n]\setminus\{k\}})\end{aligned}$$

For simplicity, we let $S = [n]\setminus\{k\}$. Now, for random $\{\Delta_i\}_{i\in[n]}, \{D_i\}_{i\in[n]}, b, \{r_i\}_{i\in S}$ where $D_i = \Delta'_i bs_i$, and $(E_m, \{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in S}, \{E_i^\gamma\}_{i\in S}, E_k^\beta, E_k^\gamma, E^{\mathsf{Check}}, l)$ are the elements output from $\mathbf{Hyb}_1$, we have

$$\begin{aligned}&(E_m, \{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in S}, \{E_i^\gamma\}_{i\in S}, E_k^\beta, E_k^\gamma, E^{\mathsf{Check}}, l)\\ =&(E_m, \{\Delta_iE_i\}_{i\in[n]}, \{\Delta_i(D_ib^{-1}E)\}_{i\in S}, \{\Delta_i(D_iE)\}_{i\in S}, \Delta_kD_k(b^{-1}E),\\ &\quad \Delta_kD_kE, \Delta_kD_kE_m, \Delta_kD_k)\\ \approx_c&(E_m, \{\Delta_iE_i\}_{i\in[n]}, \{\Delta_i(D_ib^{-1}E)\}_{i\in S}, \{\Delta_i(r_iE)\}_{i\in S}, \Delta_kD_k(b^{-1}E),\\ &\quad \Delta_kD_kE, \Delta_kD_kE_m, \Delta_kD_k)\\ =&(E_m, \{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in S}, \{r'_iE_i\}_{i\in S}, E_k^\beta, E_k^\gamma, E^{\mathsf{Check}}, l)\end{aligned}$$

Finally we let $r'_k = \Delta_kD_k$, which is obviously independent from all other r'_i, then $(E_k^\beta, E_k^\gamma, E^{\mathsf{Check}}, l) = (r'_kb^{-1}E, r'_kE, r'_kE_m, r'_k)$. Note that r'_kb^{-1} gives fresh randomness since b is now independent from all other elements in RHS. Thus RHS perfectly fits the distribution for $\mathbf{Hyb}_2$. This concludes that $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are computationally indistinguishable.

□

Lemma 7. *For any $E_m \in X_m$, $\{E_i\}_{i\in[n]} \in X^n$, and s_{k_0}, s_{k_1} s.t. both $(\{E_i\}_{i\in[n]}, s_{k_0})$, $(\{E_i\}_{i\in[n]}, s_{k_1}) \in R_n$ then*

$$\mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_0}) = \mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_1})$$

Proof. We always have $\mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_0}) = \mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_1})$ for $\mathsf{ch} = 1, 2, 3$, as every elements in the output is generated independently from k. For $\mathsf{ch} = 4$, we can give a deeper look on elements in $(\mathsf{com}, \mathsf{resp}) = (E^\alpha, E^\beta, E^\gamma, E^{\mathsf{Open}}, E^{\mathsf{Check}}, l)$. The part $(E^\alpha, E^\beta, E^{\mathsf{Open}})$ is generated independent from k, and the part $(E^\gamma, E^{\mathsf{Check}}, l)$ is of the form $(\tau(\{r_i E\}_{i\in[n]}), r_k E_m, r_k)$. Since τ is a random permutation and r_i's are independent randomness, the two distributions $(\tau(\{r_i E\}_{i\in[n]}), r_{k_0} E_m, r_{k_0})$ and $(\tau(\{r_i E\}_{i\in[n]}), r_{k_1} E_m, r_{k_1})$ are obviously identical. Hence $\mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_0}) = \mathbf{Hyb}_2(E_m, \{E_i\}_{i\in[n]}, s_{k_1})$.

□

Finally, by combining Lemma 5, Lemma 6, and Lemma 7, we can directly conclude that for any $\{E_i\}_{i\in[n]} \in X^n$, and s_{k_0}, s_{k_1} s.t. $(\{E_i\}_{i\in[n]}, s_{k_0}) \in R_n$, $(\{E_i\}_{i\in[n]}, s_{k_1}) \in R_n$,

$$\left| \begin{array}{l} \Pr\limits_{(E_m, s_m) \leftarrow \mathbf{MKeygen}} [1 \leftarrow Adv^{\mathbf{Trans}(E_m, \{E_i\}_{i\in[n]}, s_{k_0})}(\mathsf{mpk})] \\ - \Pr\limits_{(E_m, s_m) \leftarrow \mathbf{MKeygen}} [1 \leftarrow Adv^{\mathbf{Trans}((E_m, \{E_i\}_{i\in[n]}, s_{k_1})}(\mathsf{mpk})] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Which immediately proves Lemma 4. This completes the proof that Σ_{GA} is an openable sigma protocol.

4 Constructing Accountable Ring Signatures

In this section, we will show how to obtain an accountable ring signature scheme from our openable sigma protocol. The construction can be decomposed into two parts. We first apply a parallel repetition to the sigma protocol for soundness amplification, then we apply the standard Fiat-Shamir transformation on the parallelized protocol to obtain the full construction. One subtle issue is that since every sigma protocol in the parallel repetition is generated independently, each parallel session of the transcript may open to a different party. Hence, we need an opening function for the parallelized protocol which returns the majority output over the opening results of the parallel sessions.

4.1 Construction

We now construct our ARS scheme $\mathcal{ARS}_{GA}$. The construction is based on our openable sigma protocol $\Sigma_{GA,\lambda}$ over the group action $\mathcal{GA}_\lambda = (G_\lambda, \mathcal{E}_\lambda)$. In our construction of Σ_{GA}, the opening relation and the base relation are both set to R_E. Thus, when transferred to an ARS scheme, we can have an identical generator for master key pairs and party key pairs, and identical key spaces $\mathcal{KP}_m = R_E = \mathcal{KP}$.

The construction of $\mathcal{ARS}_{GA}$ is detailed as follows

– **MKeygen**(1^λ):

```
1: $s_m \xleftarrow{\$} G_\lambda$
2: return $(\mathsf{mpk}, \mathsf{msk}) = (s_m E, s_m)$
```

– **Keygen**(1^λ):

```
1: $s \xleftarrow{\$} G_\lambda$
2: return $(\mathsf{pk}, \mathsf{sk}) = (sE, s)$
```

– **Sign**$(\mathsf{mpk}, S, m, \mathsf{sk})$

```
1: $t = 2\lambda|S|$
2: $\forall j \in [t], (\mathsf{com}_j, st_j) \leftarrow \Sigma_{GA}.\mathbf{Commit}(\mathsf{mpk}, S, \mathsf{sk})$
3: $(\mathsf{ch}_1, \ldots, \mathsf{ch}_t) \leftarrow H(\mathsf{com}_1, \ldots, \mathsf{com}_t, m)$
4: $\forall j \in [t], \mathsf{resp}_j \leftarrow \Sigma_{GA}.\mathbf{Resp}(\mathsf{mpk}, S, \mathsf{sk}, \mathsf{com}_j, \mathsf{ch}_j, st_j)$
5: return $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
       $:= ((\mathsf{com}_1, \ldots, \mathsf{com}_t), (\mathsf{ch}_1, \ldots, \mathsf{ch}_t), (\mathsf{resp}_1, \ldots, \mathsf{resp}_t))$
```

– **Verify**$(\mathsf{mpk}, S, m, \sigma)$:

```
1: $t = 2\lambda|S|$
2: parse $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
3: check $\overrightarrow{\mathsf{ch}} = H(\overrightarrow{\mathsf{com}}, m)$
4: check $\forall j \in [t] : 1 \leftarrow \Sigma_{GA}.\mathbf{Verify}(\mathsf{mpk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j)$
5: return 1 if all checks pass
```

– **Open**$(\mathsf{msk}, S, m, \sigma)$:

```
1: $t = 2\lambda|S|$
2: parse $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
3: $\forall j \in [t], \mathsf{out}_j \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_i)$
4: $\mathsf{pk} = \mathbf{Maj}(\{\mathsf{out}_j\}_{j\in[t]})$ {Maj outputs the majority element of a set. In
   case of ties, Maj outputs a random choice of the marjority elements.}
5: return pk
```

Theorem 3. *$\mathcal{ARS}_{GA}$ is a secure ARS scheme.*

See Section 4.2 for the proof. By applying the transformation from Section 2.6, we immediately get the following corollary.

Corollary 1. *$\mathcal{GS}^{\mathcal{ARS}_{GA}}$ is a secure group signature scheme.*

This completes our construction of *both* an accountable ring signature scheme and a group signature scheme.

Remark. One additional benefit of using class group action as the key relation is that honest public keys can be efficiently verified. As discussed in Section 2.1, any $E_i \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ is a valid public key since the group action is transitive, and furthermore any $E_i \notin \mathcal{E}\ell\ell_p(\mathcal{O}, \pi_p)$ can be efficiently detected. This prevents the possibility of malformed master key or malformed public keys, which is a potential attacking interface of an ARS scheme.

4.2 Security

For the proof of Theorem 3 we again break down the theorem to proving each security property.

Lemma 8. *$\mathcal{ARS}_{GA}$ is* ***correct****.*

Proof. For any master key pair $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{KP}_m$, any key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{KP}$, and any set of public keys S such that $\mathsf{pk} \in S$, we directly have $(\mathsf{mpk}, \mathsf{msk}) \in R_m$ and $(S, \mathsf{sk}) \in R_n$ where $n = |S|$. Let $\sigma \leftarrow \mathbf{Sign}(\mathsf{mpk}, S, m, \mathsf{sk})$ be an honest signature on message m and ring S. Notice that in an honest execution of **Sign**, each com_j and resp_j is honestly generated according to Σ_{GA}. Thus by the completeness of Σ_{GA}, we know for every $j \in [t]$ with probability $1 - \mathsf{negl}(\lambda)$, that $1 \leftarrow \Sigma_{GA}.\mathbf{Verify}(\mathsf{mpk}, S, \mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j)$ and $\mathsf{pk} \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{mpk}, S, \mathsf{com}_j)$. Hence we directly obtain that, with probability $1 - t \cdot \mathsf{negl}(\lambda) = 1 - \mathsf{negl}(\lambda)$, we have that $1 \leftarrow \mathbf{Verify}(\mathsf{mpk}, S, m, \sigma)$ and $\mathsf{pk} \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma)$. This concludes the proof that $\mathcal{ARS}_{GA}$ is correct. □

Lemma 9. *$\mathcal{ARS}_{GA}$ is* ***anonymous****, (assuming DDHAP is hard for $\mathcal{GA}$)*

Proof. The anonymity of $\mathcal{ARS}_{GA}$ follows immediately from the CWI property of Σ_{GA}. For any adversary A with at most Q queries to the random oracle, it can have at most $Q/|H| = \mathsf{negl}(\lambda)$ advantage on distinguishing $\mathbf{Sign}^*$ and $(\mathbf{Trans}^*)^t$. And by CWI from Σ_{GA}, we have $\mathbf{Trans}^*(\mathsf{mpk}, S, \mathsf{sk}_{id_0}) \approx_c \mathbf{Trans}^*(\mathsf{mpk}, S, \mathsf{sk}_{id_1})$. Hence we can directly conclude that $\mathbf{Sign}^*(\mathsf{mpk}, S, \mathsf{sk}_{id_0}) \approx_c \mathbf{Sign}^*(\mathsf{mpk}, S, \mathsf{sk}_{id_1})$, which proves that $\mathcal{ARS}_{GA}$ is anonymous. □

Lemma 10. *$\mathcal{ARS}_{GA}$ is* ***unforgeable****, (assuming GAIP is hard for $\mathcal{GA}$)*

Proof. Assume that there exists an efficient adversary A that wins $G^{\mathsf{UF}}_{A,n_h}(\mathsf{mpk}, \mathsf{msk})$ on some valid key pair $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{KP}_m$ with non-negligible probability. We aim to show that we can construct some algorithm A_{GAIP} which runs A as a subroutine and breaks $GAIP$ on $\mathcal{GA}$.

First, we replace the **Sign** oracle with a simulator, so that A_{GAIP} can emulate the oracle responses to A. We consider a modified game $G^{\mathsf{UF},1}_{A,n_h}$ which replaces the signing oracle $\mathbf{Sign}(\cdot, \cdot, \cdot, \mathsf{sk}_i \in \mathsf{Hon})$ by a simulator $\mathbf{Sim}(\cdot, \cdot, \cdot, \mathsf{pk}_i \in \mathsf{Hon})$, where **Sim** is defined as follows:

- $\mathbf{Sim}(\mathsf{mpk}, S, m, \mathsf{pk} \in S)$:

```
1: $t = 2\lambda|S|$
2: for $j \in [t]$, $(\mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j) \leftarrow \Sigma_{GA}.\mathbf{Sim}(\mathsf{mpk}, S, \mathsf{pk} \in S)$
3: program $H(\mathsf{com}_1, \dots, \mathsf{com}_t, m) := (\mathsf{ch}_1, \dots, \mathsf{ch}_t)$
4: return $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
            $:= ((\mathsf{com}_1, \dots, \mathsf{com}_t), (\mathsf{ch}_1, \dots, \mathsf{ch}_t), (\mathsf{resp}_1, \dots, \mathsf{resp}_t))$
```

Since $\Sigma_{GA}.\mathbf{Sim}$ is a perfect simulator, any adversary with $Q = poly(\lambda)$ queries to H cannot distinguish **Sign** from **Sim** with non-negligible probability. Thus A should also win $G^{\mathsf{UF},1}_{A,n_h}$ with non-negligible probability.

Now, since A wins $G^{\mathsf{UF},1}_{A,n_h}(\mathsf{mpk}, \mathsf{msk})$ only if it outputs some (R, m^*, σ^*) such that $\mathsf{out}^* \leftarrow \mathsf{Open}(\mathsf{msk}, R, m, \sigma^*)$ satisfies $\mathsf{out}^* = \mathsf{pk}_i \in \mathsf{Hon}$ or $\mathsf{out}^* = \perp$, either A wins with non-negligible probability with $\mathsf{out}^* = \perp$, or there exists some k such that A wins with non-negligible probability with $\mathsf{out}^* = \mathsf{pk}_k \in \mathsf{Hon}$. We deal with these cases separately.

We first prove that there cannot exist efficient $A_\perp$ that wins $G^{\mathsf{UF},1}_{A,n_h}(\mathsf{mpk}, \mathsf{msk})$ with non-negligible probability with $\mathsf{out}^* = \perp$. If such $A_\perp$ exists, we can construct an algorithm B that honestly generates $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [n_h]}$ and runs $A_\perp$ with input $\mathsf{Hon} = \{\mathsf{pk}_i\}_{i \in [n_h]}$. The oracle **Corrupt** can be perfectly emulated by B since B holds every sk_i. With non-negligible probability, $A_\perp$ will output valid $(S, m, \sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}}))$ such that $\perp \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma)$. By applying the improved forking lemma (Theorem 1), with $r = O(Q/\varepsilon)$ rewinds of $A_\perp$, it would, with constant probability, output four valid signatures $(S, m, \sigma^1, \dots, \sigma^4)$ with identical $\overrightarrow{\mathsf{com}}$ and pairwise distinct $\overrightarrow{\mathsf{ch}}^c$, and that $\perp \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma^c)$ for all $c \in [4]$. We now claim that with high probability, we can find some parallel session $j \in [t]$ such that $\perp \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_j)$ and $\mathsf{ch}^1_j, \dots, \mathsf{ch}^4_j$ are distinct. Note that this is not trivially true, as the forking lemma only promises that $\overrightarrow{\mathsf{ch}}^1, \dots, \overrightarrow{\mathsf{ch}}^4$ are pairwise distinct as vectors, so they might not be pairwise distinct on any index j.

Let T be the set of indices j where $\perp \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_j)$. Since $\perp \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma)$, by the definition of **Open**, $\perp$ must be (one of) the majority output among the t parallel sessions. Thus $|T| \geq t/(|S|+1) \geq \lambda$. We say that four challenges $\overrightarrow{\mathsf{ch}}'^1, \dots, \overrightarrow{\mathsf{ch}}'^4$ are **good** on T if there exists some $j \in T$ such that $\mathsf{ch}'^1_j, \dots, \mathsf{ch}'^4_j$ are distinct. For 4 independently random challenges in $[4]^t$, the probability that they are good on T is $1 - (1 - (4!/4^4))^{|T|} = 1 - \mathsf{negl}(\lambda)$.

Unfortunately, the challenges $\overrightarrow{\mathsf{ch}}^1, \dots, \overrightarrow{\mathsf{ch}}^4$ obtained from rewinding A are not necessarily independent. To cope with this, we will need the fact that in each rewind of A, the *valid* $\overrightarrow{\mathsf{ch}}$ is a new random output from the new random oracle H. Thus, the finally output 4-tuple $\overrightarrow{\mathsf{ch}}^1, \dots, \overrightarrow{\mathsf{ch}}^4$ must be a subset of $r = O(Q/\varepsilon)$ independent random samples from $[4]^t$. By the union bound, the probability that *all* 4-tuples in the r samples are good on T is $1 - \binom{r}{4}\mathsf{negl}(|T|) \geq 1 - \mathsf{negl}(\lambda)$. Thus we can find $j \in T$ such that $\mathsf{ch}^1_j, \dots, \mathsf{ch}^4_j$ are distinct with probability $1 - \mathsf{negl}(\lambda)$.

For such j, we without loss of generality let $(\mathsf{ch}_j^1, \ldots, \mathsf{ch}_j^4) = (1, \ldots, 4)$ and consider $(S, \mathsf{com}_j, \mathsf{resp}_j^1, \ldots, \mathsf{resp}_j^4)$. Now B achieves $\forall c \in [4], 1 \leftarrow \Sigma_{GA}.\mathbf{Verify}(S, \mathsf{com}_j, c, \mathsf{resp}_j^c)$, and $\perp \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_j)$. Thus B violates the 4-special soundness property of Σ_{GA} and brings a contradiction. Hence such $A_\perp$ cannot exist.

Now we consider the case where some A_k wins $G_{A,n_h}^{\mathsf{UF},1}(\mathsf{mpk}, \mathsf{msk})$ with non-negligible probability with $\mathsf{out}^* = \mathsf{pk}_k$. We will show that if such A_k exists, we can build A_{GAIP} from A_k.

We first do some modification on the **Corrupt** oracle by considering the game $G_{A,n_j,k}^{\mathsf{UF},2}(\mathsf{mpk}, \mathsf{msk}, \mathsf{pk})$ which, on top of $G_{A,n_h}^{\mathsf{UF},1}$, applies the following modification:

1. Set $(\mathsf{pk}_k, \mathsf{sk}_k) = (\mathsf{pk}, \perp)$. Other $(\mathsf{pk}_i, \mathsf{sk}_i)$'s are generated honestly for $i \in [n_h] \setminus \{k\}$
2. Replace the oracle **Corrupt**() with $\mathbf{Corrupt}_k^*()$, which aborts when pk_k is queried and otherwise honestly outputs as **Corrupt**.
3. Change the winning condition to: A wins if (m^*, σ^*) is not an output of **Sign**, $1 \leftarrow \mathbf{Verify}(\mathsf{mpk}, S, m^*, \sigma^*)$ and $\mathsf{pk} = \mathsf{pk}_k \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma^*)$. In other words, we restrict to the case that A wins with $\mathsf{out}^* = \mathsf{pk}_k$.

Note that for A_k to win $G_{A,n_h}^{\mathsf{UF},1}(\mathsf{mpk}, \mathsf{msk})$ with $\mathsf{out}^* = \mathsf{pk}_k$, it cannot query $\mathbf{Corrupt}(\mathsf{pk}_k)$, thus A_k should also win $G_{A,2,k}^{\mathsf{UF}}(\mathsf{mpk}, \mathsf{msk}, \mathsf{pk})$ with non-negligible probability, $1/n_h$ times as likely, for pk honestly generated from **Keygen**. By the construction of **Keygen**, it is equivalent to sampling a random $E_{\mathsf{ch}} \in \mathcal{E}$.

Now, for A_k winning $G_{A,n_h,k}^{\mathsf{UF},2}(\mathsf{mpk}, \mathsf{msk}, \mathsf{pk})$ with non-negligible probability, we can similarly construct a algorithm B that honestly generates $n_h - 1$ key pairs $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [n_h] \setminus \{k\}}$ and runs A_k with input $\mathsf{Hon} = \{\mathsf{pk}_i\}_{i \in [n_h]}$ where $\mathsf{pk}_k = \mathsf{pk}$. Then again by applying the improved forking lemma, with the same probability, $r = O(Q/\varepsilon)$ rewinds of A_k will output four valid signatures $(S, m, \sigma^1, \ldots, \sigma^4)$ with identical $\overrightarrow{\mathsf{com}}$ and pairwise distinct $\overrightarrow{\mathsf{ch}}^c$, so that $\mathsf{pk} \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma^c)$ for all $c \in [4]$. Again by the same argument as in the case of $A_\perp$, we can with high probability find some $j \in [t]$ such that $\mathsf{pk} \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_j)$ and $\mathsf{ch}_j^1, \ldots, \mathsf{ch}_j^4$ are distinct.

Now, without loss of generality let $(\mathsf{ch}_j^1, \ldots, \mathsf{ch}_j^4) = (1, \ldots, 4)$ and consider $(S, \mathsf{com}_j, \mathsf{resp}_j^1, \ldots, \mathsf{resp}_j^4)$. We have $\forall c \in [4], 1 \leftarrow \Sigma_{GA}.\mathbf{Verify}(S, \mathsf{com}_j, c, \mathsf{resp}_j^c)$, and that the challenge statement $\mathsf{pk} \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_j)$. Thus by the 4-special soundness property of Σ_{GA}, we can extract the matching secret key $\mathsf{sk} \leftarrow \Sigma_{GA}.\mathbf{Ext}(S, \mathsf{com}_j, \mathsf{resp}_j^1, \ldots, \mathsf{resp}_j^4)$, such that $(\mathsf{pk}, \mathsf{sk}) \in R_E$, i.e. $\mathsf{pk} = \mathsf{sk}E$.

From the previous arguments, we see that if such efficient A_k exists, then we can obtain an algorithm B based on A_k that, on inputting random $E_{\mathsf{ch}} \in \mathcal{E}$, output s such that $sE = E_{\mathsf{ch}}$ with non-negligible probability. To break GAIP, when obtaining random challenges $E_1, E_2 \in \mathcal{E}$, we simply run B twice to get s_1, s_2 such that $s_1 E = E_1, s_2 E = E_2$, then we directly obtain $E_2 = (s_1 s_2^{-1}) E_1$. Thus, we successfully construct a $GAIP$ breaker from adversary A that wins the unforgeability game, which concludes the proof that our $\mathcal{ARS}_{GA}$ is unforgeable assuming GAIP is hard for $\mathcal{GA}$. □

5 Discussion

In this work, our signature schemes are constructed under the random oracle model. Primitives under ROM from post-quantum assumptions are sometimes heuristically assumed to be post-quantum secure. Nevertheless, solid analysis under the quantum random oracle model (QROM), in which such oracles can be queried in superposition, is often preferred. As in the case of the CSI-FiSh protocol [5], our openable sigma protocol satisfies the *perfect unique response* property, which is the key property for achieving QROM security according to [16, 36]. Unfortunately, as we have introduced new components to our sigma protocol, existing results cannot be directly applied. We hence leave the QROM analysis for our protocol as an open problem for future work.

In our setting, we have premised an honest manager, as the opening result is only available to the manager. A corrupted manager can thus incriminate any party as the signer of an arbitrary signature. To cope with this, many previous works on group signatures provide an extra *judging function*, which allows the manager to generate a publicly verifiable proof for its opening results. Due to the majority voting that we have adopted in our opening design, we do not know yet how to construct a proof for the exact opening output. Nevertheless, we are able to provide, see Appendix A, a weaker variant where the manager proves the following: a sufficient number of sessions within a signature is opened to the claimed signer k. This is essentially proving for multiple sessions that $s_m E_k^\beta = E^{\mathsf{Open}}$ (as in Section 3.3), which is done with a slight twist to Couveignes' sigma protocol. Though weaker, this notion is still meaningful as it also prevents a corrupted manager from incriminating honest non-signers. We will leave the construction supporting a full-fledged judging function to future work.

Acknowledgments

Mi-Ying Huang would like to thank her current advisor Jiapeng Zhang. Authors were supported by Taiwan Ministry of Science and Technology Grant 109-2221-E-001-009-MY3, Sinica Investigator Award (AS-IA-109-M01), Executive Yuan Data Safety and Talent Cultivation Project (AS-KPQ-109-DSTCP), and Young Scholar Fellowship (Einstein Program) of the Ministry of Science and Technology (MOST) in Taiwan, under grant number MOST 110-2636-E-002-012, and by the Netherlands Organisation for Scientific Research (NWO) under grants 628.001.028 (FASOR) and 613.009.144 (Quantum Cryptanalysis of Post-Quantum Cryptography). This work was carried out while the fifth author was visiting Academia Sinica, she is grateful for the hospitality.

References

1. Rachid El Bansarkhani and Rafael Misoczki. G-Merkle: A Hash-Based Group Signature Scheme from Standard Assumptions. In *PQCrypto*, volume 10786 of *Lecture Notes in Computer Science*, pages 441–463. Springer, 2018.

2. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
3. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
4. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492. Springer, 2020.
5. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations. In *ASIACRYPT (1)*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019.
6. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
7. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of Fully Dynamic Group Signatures. In *ACNS*, volume 9696 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2016.
8. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short Accountable Ring Signatures Based on DDH. In *ESORICS (1)*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265. Springer, 2015.
9. Emmanuel Bresson and Jacques Stern. Efficient Revocation in Group Signatures. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 190–206. Springer, 2001.
10. Ernest F. Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2000.
11. Jan Camenisch and Markus Michels. A Group Signature Scheme with Improved Efficiency. In *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 1998.
12. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An Efficient Post-Quantum Commutative Group Action. In *ASIACRYPT (3)*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
13. Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the Decisional Diffie-Hellman Problem for Class Group Actions Using Genus Theory. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 92–120. Springer, 2020.
14. David Chaum and Eugène van Heyst. Group Signatures. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
15. Jean-Marc Couveignes. Hard Homogeneous Spaces. Cryptology ePrint Archive, Report 2006/291, 2006.
16. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.

17. Ali El Kaafarani, Shuichi Katsumata, and Federico Pintore. Lossy CSI-FiSh: Efficient Signature Scheme with Tight Reduction to Decisional CSIDH-512. In *Public Key Cryptography (2)*, volume 12111 of *Lecture Notes in Computer Science*, pages 157–186. Springer, 2020.
18. Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A Provably Secure Group Signature Scheme from Code-Based Assumptions. In *ASIACRYPT (1)*, volume 9452 of *Lecture Notes in Computer Science*, pages 260–285. Springer, 2015.
19. Luca De Feo and Steven D. Galbraith. SeaSign: Compact Isogeny Signatures from Class Group Actions. In *EUROCRYPT (3)*, volume 11478 of *Lecture Notes in Computer Science*, pages 759–789. Springer, 2019.
20. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
21. S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A Group Signature Scheme from Lattice Assumptions. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2010.
22. Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *Int. J. Secur. Networks*, 1(1/2):24–45, 2006.
23. Sudhakar Kumawat and Souradyuti Paul. A New Constant-Size Accountable Ring Signature Scheme Without Random Oracles. In *Inscrypt*, volume 10726 of *Lecture Notes in Computer Science*, pages 157–179. Springer, 2017.
24. Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-Based Group Signatures with Logarithmic Signature Size. In *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.
25. Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient Sanitizable Signatures Without Random Oracles. In *ESORICS (1)*, volume 9878 of *Lecture Notes in Computer Science*, pages 363–380. Springer, 2016.
26. Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions. In *ASIACRYPT (2)*, volume 10032 of *Lecture Notes in Computer Science*, pages 373–403, 2016.
27. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-Based Group Signatures: Achieving Full Dynamicity with Ease. In *ACNS*, volume 10355 of *Lecture Notes in Computer Science*, pages 293–312. Springer, 2017.
28. Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.
29. Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler Efficient Group Signatures from Lattices. In *Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 401–426. Springer, 2015.
30. David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.
31. David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptol.*, 13(3):361–396, 2000.
32. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
33. Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptol.*, 4(3):161–174, 1991.

34. Dawn Xiaodong Song. Practical forward secure group signature schemes. In *CCS*, pages 225–234. ACM, 2001.
35. Anton Stolbunov. *Cryptographic Schemes Based on Isogenies.* PhD thesis, 01 2012.
36. Dominique Unruh. Quantum Proofs of Knowledge. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2012.
37. Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.
38. Shouhuai Xu and Moti Yung. Accountable Ring Signatures: A Smart Card Approach. In *CARDIS*, volume 153 of *IFIP*, pages 271–286. Kluwer/Springer, 2004.

A Judging the opening

As a natural byproduct of our construction, we could also empower the manager to generate a proof π additionally from **Open** that could be publicly verified using an additional algorithm **Judge** as (re)defined below:

- **Open**$(\mathsf{msk}, S = \{\mathsf{pk}_i\}_{i\in[n]}, m, \sigma) \to (\mathsf{pk}, \pi) \in (S \cup \{\bot\}) \times \{0,1\}^*$: The redefined open algorithm not only reveals signer identity pk but also produces a publicly verifiable proof π for it.
- **Judge**$(\mathsf{mpk}, S = \{\mathsf{pk}_i\}_{i\in[n]}, \sigma, \mathsf{pk}, \pi) \to \mathsf{acc} \in \{0,1\}$: The judge algorithm accepts if the manager opened correctly,

Note that in Section 3.3, the opening within the sigma protocol is done by picking the index k such that $s_m E_k^\beta = E^{\mathsf{Open}}$. A manager could therefore prove this equality in a Schnorr-like manner, re-starting from the sigma protocol Σ_{GA} with three additional algorithms **JCommit**, **JResp**, **JVerify**.

- **JCommit**$(s_m := \mathsf{msk}, \{E_i\}_{i\in[n]} := \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{com})$:

```
1: $b' \xleftarrow{\$} G$
2: parse $\mathsf{com} = (\{\mathsf{E}_i^\alpha\}_{i\in[n]}, \{\mathsf{E}_i^\beta\}_{i\in[n]}, \tau(\{\mathsf{E}_i^\gamma\}_{i\in[n]}), \mathsf{E}^{\mathsf{Open}}, \mathsf{E}^{\mathsf{Check}})$ {We use $\tau(\cdot)$ as a lazy convention of sending a permuted list}
3: $E^{\mathsf{Judge}} := b' E^{\mathsf{Open}}$
4: $E_m^{b'} := b' s_m E$
5: return $(\mathsf{jcom}, \mathsf{jst}) = \Big((E^{\mathsf{Judge}}, E_m^{b'}), (b', s_m)\Big)$
```

- **JResp**$(E_m, \{E_i\}_{i\in[n]}, \mathsf{jcom}, \mathsf{jch}, \mathsf{jst})$:

```
1: parse $\mathsf{jst} = (b', s_m)$
2: if $\mathsf{jch} = 0$ then
3:     return $\mathsf{jresp} := b'$
4: if $\mathsf{jch} = 1$ then
5:     return $\mathsf{jresp} := l' = b' s_m$
```

- **JVerify**$(E_m := \mathsf{mpk}, \{E_i\}_{i\in[n]} := \{\mathsf{pk}_i\}_{i\in[n]}, E_k := \mathsf{pk}, \mathsf{com}, \mathsf{jcom}, \mathsf{jch}, \mathsf{jresp})$:

```
1: parse $\mathsf{com} = (\{E_i^\alpha\}_{i\in[n]}, \{E_i^\beta\}_{i\in[n]}, \tau(\{E_i^\gamma\}_{i\in[n]}), E^{\mathsf{Open}}, E^{\mathsf{Check}})$
2: parse $\mathsf{jcom} = (E^{\mathsf{Judge}}, E_m^{b'})$
```

```
3: if $\mathsf{jch} = 0$ then
4:     check $E^{\mathsf{Judge}} = b'E^{\mathsf{Open}}$
5:     check $E_m^{b'} = b'E_m$
6: if $\mathsf{jch} = 1$ then
7:     check $E^{\mathsf{Judge}} = l'E_k^{\beta}$
8:     check $E_m^{b'} = l'E$
9: return 1 if all check pass
```

For each run of **Commit** $\to (\mathsf{com}, \mathsf{st})$, we have to do additionally ι repetitions of **JCommit** (and thus ιt repetitions in total) to confirm that it is opened to the k-th signer with $\mathsf{negl}(\iota)$ probability. Similar as before, the Fiat-Shamir transform is applied for non-interactivity as follows.

– **Open**$(\mathsf{msk}, S = \{\mathsf{pk}_i\}_{i\in[n]}, m, \sigma)$

```
1: $t = 2\lambda|S|$; $\iota = \lambda$
2: parse $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
3: $\forall j \in [t], \mathsf{out}_j \leftarrow \Sigma_{GA}.\mathbf{Open}(\mathsf{msk}, S, \mathsf{com}_i)$
4: $\forall (i,j) \in [\iota] \times [t], \mathsf{jcom}_{i,j} \leftarrow \Sigma_{GA}.\mathbf{JCommit}(\mathsf{msk}, \{E_i\}_{i\in[n]}, \mathsf{com}_j)$
5: $\forall (i,j) \in [\iota] \times [t], \mathsf{ch}_{i,j} \leftarrow H(\sigma, \{\mathsf{jcom}_{i,j}\}_{(i,j)\in[\iota]\times[t]})$
6: $\forall (i,j) \in [\iota]\times[t], (\mathsf{jresp}_{i,j}, \mathsf{jst}_{i,j}) \leftarrow \Sigma_{GA}.\mathbf{JResp}(E_m, \{E_i\}_{i\in[n]}, \mathsf{jcom}_{i,j}, \mathsf{jch}_{i,j}, \mathsf{jst}_{i,j})$
7: $\mathsf{pk} = \mathbf{Maj}(\{\mathsf{out}_j\}_{j\in[t]})$ {Maj outputs the majority element of a set. In
   case of ties, Maj outputs a random choice of the marjority elements.}
8: $\pi := \{\mathsf{jcom}_{i,j}, \mathsf{jch}_{i,j}, \mathsf{jresp}_{i,j}\}_{(i,j)\in[\iota]\times[t]}$
9: return $(\mathsf{pk}, \pi)$
```

– **Judge**$(\mathsf{mpk}, S = \{\mathsf{pk}_i\}_{i\in[n]}, \sigma, \mathsf{pk}, \pi)$:

```
1: return 0 if $\mathsf{pk} = \perp$
2: $t = 2\lambda|S|$; $\iota = \lambda$
3: parse $\sigma = (\overrightarrow{\mathsf{com}}, \overrightarrow{\mathsf{ch}}, \overrightarrow{\mathsf{resp}})$
4: parse $\pi = \{\mathsf{jcom}_{i,j}, \mathsf{jch}_{i,j}, \mathsf{jresp}_{i,j}\}_{(i,j)\in[\iota]\times[t]}$
5: $\forall j \in [t], \mathsf{jout}_j \leftarrow \bigwedge_{i\in[\iota]} \Sigma_{GA}.\mathbf{JVerify}(\mathsf{mpk}, \{\mathsf{E}_i\}_{i\in[n]}, \mathsf{pk}, \mathsf{com}_j, \mathsf{jcom}_{i,j}, \mathsf{jch}_{i,j}, \mathsf{jresp}_{i,j})$
6: return 1 if $\sum_{j\in[t]} \mathsf{jout}_j \geq \lambda$
```

Here, a corrupted manager get to selectively generate a partial proof $\{E_{\mathsf{out}_j}, \mathsf{jcom}_{i,j}, \mathsf{jch}_{i,j}, \mathsf{jresp}_{i,j}\}_{(i,j)\in[\iota]\times\mathcal{J}}$ where $\mathcal{J} \subseteq [t]$ is adaptively chosen. So long as we have $\sum_{j\in\mathcal{J}} \mathsf{jout}_j \geq \lambda$, the judged proof is accepted. This does not prevent the manager from generating accepted proofs that open to different members when $\#\{E_{\mathsf{out}_j}\} > 1$, which could happen if the corresponding signature is generated by multiple colluding signers. Otherwise, incriminating an honest non-signer would require to make up at least λ valid sessions of **Commit**, which will succeed with only negligible probability, i.e. for any PPT adversary A, any $n_h \leq n \leq \mathsf{poly}(\lambda)$ and valid master key pair $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{KP}_m$,

$$\Pr[A \text{ wins } G^{\mathsf{JUF}}_{A,n_h}] \leq \mathsf{negl}(\lambda),$$

where the *judging unforgeability game* G^{JUF}_{A,n_h} is as specified below.

G^{JUF}_{A,n_h}: Judging unforgeability game

```
1: $\forall i \in [n_h], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathbf{Keygen}(1^\lambda)$. Let $\mathsf{Hon} = \{\mathsf{pk}_i\}_{i\in[n_h]}, \mathsf{Cor} = \{\}$.
2: $(S, m^*, \sigma^*) \leftarrow A^{\mathbf{Sign}(\cdot,\cdot,\cdot,\mathsf{sk}_i \in \mathsf{Hon}),\mathbf{Corrupt}(\cdot)}(\mathsf{Hon})$
   $\{\mathbf{Corrupt}(\mathsf{pk}_i)$ returns $\mathsf{sk}_i$ for $\mathsf{pk}_i \in \mathsf{Hon}$ and stores query $\mathsf{pk}_i$ in list $\mathsf{Cor}\}$
3: $A$ wins if $(m^*, \sigma^*)$ is not an output of Sign, $1 \leftarrow \mathbf{Verify}(\mathsf{mpk}, S, m^*, \sigma^*)$,
   $(\mathsf{pk}, \pi) \leftarrow \mathbf{Open}(\mathsf{msk}, S, m, \sigma^*)$ satisfies $\mathsf{pk} \in \{\perp\} \cup \mathsf{Hon} \setminus \mathsf{Cor}$, and $1 \leftarrow$
   $\mathbf{Judge}(\mathsf{mpk}, S, \sigma, \mathsf{pk}, \pi)$.
```