Malleable Commitments from Group Actions and Zero-Knowledge Proofs for Circuits based on Isogenies

Mingjie Chen[1], Yi-Fu Lai[2,3], Abel Laval[4], Laurane Marco[5], and Christophe Petit[1,4]

[1] University of Birmingham, UK
[2] University of Auckland, New Zealand
[3] Ruhr-University Bochum, Germany
[4] Université Libre de Bruxelles, Belgium
[5] EPFL, Switzerland

Keywords: group action, isogeny-based cryptography, commitments, generic zero-knowledge proof of knowledge, post-quantum cryptography

Abstract Zero-knowledge proofs for NP statements are an essential tool for building various cryptographic primitives and have been extensively studied in recent years. In a seminal result from Goldreich, Micali and Wigderson [17], zero-knowledge proofs for NP statements can be built from any one-way function, but this construction leads very inefficient proofs. To yield practical constructions, one often uses the additional structure provided by homomorphic commitments.

In this paper, we introduce a relaxed notion of homomorphic commitments, called *malleable commitments*, which requires less structure to be instantiated. We provide a malleable commitment construction from the ElGamal-type isogeny-based group action from Eurocrypt'22 [5]. We show how malleable commitments with a group structure in the malleability can be used to build zero-knowledge proofs for NP statements, improving on the naive construction from one-way functions. We consider three representations: arithmetic circuits, rank-1 constraint systems and branching programs. This work gives the first attempt at constructing a post-quantum generic proof system from isogeny assumptions (the group action DDH problem). Though the resulting proof systems are linear in the circuit size, they possess interesting features such as non-interactivity, statistical zero-knowledge, and online-extractability.

1 Introduction

Many cryptographic applications such as multiparty computations, verifiable computations, cryptocurrencies and more, rely on zero-knowledge proofs as an underlying primitive. In 1991, Goldreich, Micali and Wigderson [17] introduced a construction of proof systems for all NP statements from one-way functions. However, such proofs remain inefficient in general and in practice, one usually relies on the structure given by *homomorphic* commitments to build efficient zero-knowledge proofs [20,10,11].

Current homomorphic commitment constructions rely on classical hardness assumptions such as the hardness of the discrete logarithm problem [23] and factorisation or pairing-based assumptions [19]. The security of these problems no longer holds in a post-quantum setting. Lattice-based cryptography provides some constructions of post-quantum commitments with some homomorphic properties [16,18,9]. However, due to the weaker algebraic structure offered by isogeny-based cryptography, having homomorphic commitments does not seem plausible. Hence, this brings us to the main question of this work:

> *Can we construct a zero-knowledge proof for generic NP statements from isogenies better than from one-way functions?*

Our contributions. In this work, we build a generic zero-knowledge proof of knowledge from isogeny-based assumptions. More specifically, we work under the known-order effective group action model [1] which can be instantiated from isogenies [7,13]. We consider two representations for NP statements: arithmetic circuits over a small field and matrix-based branching programs over a large field, where the latter can serve as an efficient representation for a shallow circuit. We construct proof systems for proving the addition and multiplication gates for the arithmetic circuits over a small field or $\mathbb{Z}_h$ for some small $h \in \mathbb{N}$. Then, using these two components we construct a proof system for the rank-1 constraints system over a small field or ring, which is a more efficient representation for an arithmetic circuit. Furthermore, we construct a proof system for a matrix branching relation. Using the technique given in [5], each of these constructions leads to an online-extractable zero-knowledge proof of knowledge for NP statements without trusted setup under the random oracle model.

We use the known-order effective group action (KO-EGA) model to develop our schemes, where we know the group order and have an efficient algorithm for computing the action of ng for any g within the (additive) group and $n \in \mathbb{N}$. Currently, the only two known post-quantum instantiations of KO-EGA are from isogeny-based group actions, specifically CSI-FiSh and SCALLOP [7,13], and the quantum security levels of these two instantiations are somewhat lower than the NIST-1 requirement [24]. As such, our results remain predominantly theoretical. However, it is essential to highlight that the KO-EGA model implies the existence of numerous cryptographic primitives with significant relevance in real-world applications [6,14,5,21]. The challenge of instantiating a stronger KO-EGA remains an open problem and an active area of research, which is orthogonal to the focus of our paper. Our primary focus centers on constructing schemes within this model and providing performance metrics in terms of the number of group actions, group elements, and set elements.

Another limitation of our proof systems is that their sizes grow linearly with the input length. We describe the obstacles and challenges towards obtaining succinct proofs in Sec. 6.4.

Technical Overview. In this paper, we first introduce *malleable commitments*, a generalization of homomorphic commitments. Intuitively, a malleable commitment scheme for a given relation must satisfy the following property: given a commitment com on a message m, there exists an efficient algorithm that can derive a commitment com' on m' such that m and m' are related. The idea is to settle for a middle point between the absence of structure given by commitments from one-way functions, and the constraints of homomorphic commitments.

Our construction of malleable commitment is based on an adaptation of the public key encryption scheme of [5] (Sec. 4). It is computationally hiding and perfectly binding, both classically and in a post-quantum setting. We develop ad hoc proof systems for proving the correctness of a given commitment, taking advantage of the malleability property for greater efficiency (Sec. 5).

Finally, we use these proof systems as fundamental tools to provide zero-knowledge proofs for NP statements (Sec. 6). More precisely, we derive proof systems for arithmetic circuits, rank-1 constraint systems and branching programs. The constructions for arithmetic circuits and rank-1 constraint systems impose some size restrictions on the underlying field as well as the message space, which can be lifted using the branching program approach.

2 Preliminaries

We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible, written $\mathsf{negl}(\lambda)$ if its absolute value is asymptotically dominated by $\mathcal{O}(x^{-n})$ for all $n > 0$. We write PPT for probabilistic polynomial time algorithm. For $n \in \mathbb{N}$, we let $[n] = \{1, \cdots, n\}$.

2.1 Commitment Scheme

Definition 1 (Commitment scheme). *A commitment scheme is , Given a security parameter λ, the algorithms* $\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}$ *are defined in the following way:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: A probabilistic polynomial time algorithm that takes as input the security parameter λ and outputs the public parameters* pp*. The public parameters* pp *are implicitly given as input to the following two algorithms.*
- $\mathsf{Commit}(m, r) \to \mathsf{com}$*: A probabilistic polynomial time algorithm that takes a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ and outputs a commitment* $\mathsf{com} \in \mathcal{C}$*.*
- $\mathsf{Verify}(\mathsf{com}, (m, r)) \to 0/1$*: A deterministic algorithm that takes as input a commitment value* com *and a message-randomness pair (m, r) and returns* 1 *if* $\mathsf{Commit}(m, r) = \mathsf{com}$*, 0 otherwise.*

The security of a commitment scheme is characterized by two properties: it must be *hiding* and *binding*. Hiding guarantees that adversary should not be able to recover the original committed message from seeing the commitment, and binding guarantees that an adversary cannot open a commitment to two distinct values.

Definition 2 (Hiding security). *A commitment scheme is* hiding *if for any probabilistic polynomial time adversary $\mathcal{A}$ playing the* Hide_b *game the advantage is negligible i.e.*

$$\mathsf{Adv}^{\mathsf{Hide}}(\mathcal{A}) = |\Pr[\mathsf{Hide}_1(\mathcal{A}) \to 1] - \Pr[\mathsf{Hide}_0(\mathcal{A}) \to 1]| = \mathsf{negl}(\lambda)$$

where the Hide_b *game is defined as follows: given public parameters* pp*,* $\mathcal{A}$ *outputs two challenge messages* m_0, m_1 *and a state* st*. The challenger samples some randomness* r*, and then computes* $\mathsf{com} = \mathsf{Commit}(m_b, r)$*. On input* st *and* com*,* $\mathcal{A}$ *returns a bit* b' *corresponding to its guess for the value of* b*.*

Definition 3 (Binding security). *A commitment scheme is binding if for any probabilistic polynomial time adversary* $\mathcal{A}$ *playing the* Bind *game, the advantage is negligible,i.e.,*

$$\mathsf{Adv}^{\mathsf{Bind}}(\mathcal{A}) = \Pr[\mathsf{Bind}(\mathcal{A}) \to 1] = \mathsf{negl}(\lambda)$$

where the Bind *game is defined as follows: Given the public parameters* pp*,* $\mathcal{A}$ *must output two* different *messages* m_0, m_1 *and associated randomness* r_0, r_1 *and wins if* $\mathsf{Commit}(m_0, r_0) = \mathsf{Commit}(m_1, r_1)$*.*

2.2 Group Actions

Definition 4 (Group Action). *A group* G*, with group operation* $+$*, is said to act on a set* $\mathcal{E}$ *if there is a map* $\star : G \times \mathcal{E} \to \mathcal{E}$ *that satisfies the following properties:*

1. *(Identity.) If* 0 *is the identity element of* G*, then for any* $E \in \mathcal{E}$*, we have* $0 \star E = E$*.*
2. *(Compatibility.) For any* $g, h \in G$ *and any* $E \in \mathcal{E}$*, we have* $(g + h) \star E = g \star (h \star E)$*.*

Furthermore, this action is called regular *if*

3. *For every* $x_1, x_2 \in \mathcal{E}$ *there exists* $g \in G$ *such that* $x_2 = g \star x_1$*.*
4. *For* $g \in G$*,* g *is the identity element if and only if there exists some* $x \in \mathcal{E}$ *such that* $x = g \star x$*.*

In order to use group actions to build our primitives, we require some efficient (PPT) algorithms. We adopt the *known-order effective group action* (KO-EGA) framework introduced in [1]. As discussed in Sec. 1, post-quantum instantiations of KO-EGA are only known from isogenies. Obtaining a stronger parameter set from isogenies is an active research area [7,13]. In this model, there exists various constructions, such as logarithmic (linkable) ring signatures [6], threshold signatures [14], logarithmic (and tightly secure) group signatures [5], and compact blind signatures [21].

Definition 5 (Known-order Effective Group Action). *Let* $(G, \mathcal{E}, \star)$ *be a group action. For an* $E_0 \in \mathcal{E}$*, we say* $(G, \mathcal{E}, \star, E_0)$ *forms a* known-order effective *group action if the following properties are satisfied:*

1. *The group* G *is finite and both the structure of* $G \cong \bigoplus_i \mathbb{Z}/m_i\mathbb{Z}$ *and a minimal generating set* $\langle g_i \rangle = G$ *are known.*
2. *The set* $\mathcal{E}$ *is finite and there exist* PPT*algorithms for membership testing and for generating a unique bit-string representation for every element in* $\mathcal{E}$*.*
3. *There exists a distinguished element* $E_0 \in \mathcal{E}$ *for which the bit-string representation is publicly known.*

4. *There exists a* PPT*algorithm to evaluate* $ng_i \star E$ *for any natural number* $n \in \{1, ..., \prod_i m_i\}$*,* $E \in \mathcal{E}$ *and* g_i *from the generating set.*

Since the group order of G is given, one can sample uniformly at random from G. Similarly, by multiplying by a proper factor, one can obtain a generating set for any subgroup of G and sample uniformly at random from the subgroup. Throughout this paper, we consider a regular known-order effective group action from an abelian group. Hard problems in the context of group actions extend naturally from their classical counterpart, namely we have an analog of the discrete logarithm problem called group-action inverse problem (GAIP) as well as analogous of the decisional Diffie-Hellman problems for group actions.

Definition 6 (Group Action Inverse Problem). *Let* $(G, \mathcal{E}, \star, E_0)$ *be a group action with a distinguished element* $E_0 \in \mathcal{E}$*. Given* E *sampled uniformly in* $\mathcal{E}$*, the* Group Action Inverse Problem (GAIP) *consists in finding an element* $g \in G$ *such that* $g \star E_0 = E$*.*

Definition 7 (Decisional Diffie-Hellman Problem). *Let* $(G, E_0, \mathcal{E}, \star)$ *be a group action. The challenger generates* $s_1, s_2, s_3 \in G$ *and* $b \in \{0, 1\}$*, and gives* $(s_1 \star E_0, s_2 \star E_0, ((1-b)(s_1+s_2)+b(s_3)) \star E_0)$ *to the adversary* $\mathcal{A}$*. Then, Decisional Diffie-Hellman (DDH) problem consists in* $\mathcal{A}$ *returning* $b' \in \{0, 1\}$ *to guess* b*. The adversary* $\mathcal{A}$ *wins if* $b = b'$*. We define the advantage of* $\mathcal{A}$ *to be* $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) := |\Pr[\mathcal{A}\ wins.] - 1/2|$ *where the probability is taken over the randomness of* s_1, s_2, s_3, b*.*

2.3 Sigma Protocols

Definition 8 (Sigma Protocol). *A sigma protocol* Π_Σ *is a three-move proof system for a relation* R *consisting of oracle-calling* PPT*algorithms* $(P = (P_1, P_2)$*,* $V = (V_1, V_2))$*, where* V_2 *is deterministic. We assume* P_1 *and* P_2 *share states and so do* V_1 *and* V_2*. Let* ChSet *be the challenge set.* Π_Σ *proceeds as follows:*

- *The prover, on input* $(\mathsf{st}, \mathsf{wt}) \in R$*, runs* $\mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})$ *and sends a* commitment com *to the verifier.*
- *The verifier runs* $\mathsf{chall} \leftarrow V_1^{\mathcal{O}}(1^\lambda)$*, drawing a random challenge from* ChSet*, and sends it to the prover.*
- *The prover, given* chall*, runs* $\mathsf{resp} \leftarrow P_2^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, \mathsf{chall})$ *and returns a* response resp *to the verifier.*
- *The verifier runs* $V_2^{\mathcal{O}}(\mathsf{st}, \mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *and outputs* $\top$*(accept) or* $\bot$*(reject).*

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

Definition 9 (Correctness). *A sigma protocol* Π_Σ *is said to be* correct *if for all* $\lambda \in \mathbb{N}$*,* $(\mathsf{st}, \mathsf{wt}) \in R$*, given that the prover and the verifier both follow the protocol specifications, then the verifier always outputs* $\top$*.*

Definition 10 (High Min-Entropy). *We say a sigma protocol $\mathit{\Pi}_\Sigma$ has $\alpha(\lambda)$* min-entropy *if for any $\lambda \in \mathbb{N}$, $(\mathsf{st}, \mathsf{wt}) \in R$, and a possibly computationally-unbounded adversary $\mathcal{A}$, we have*

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

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

Additionally, we require a sigma protocol to be honest verifier zero-knowledge (HVZK) as well as 2-special sound.

Definition 11 (Honest Verifier Zero-Knowledge (HVZK)). *We say $\mathit{\Pi}_\Sigma$ is* honest-verifier-zero-knowledge *for relation R if there exists a* PPT*simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle $\mathcal{O}$ such that for any statement-witness pair $(\mathsf{st}, \mathsf{wt}) \in R$, $\mathsf{chall} \in \mathsf{ChSet}$, $\lambda \in \mathbb{N}$ and any computationally-unbounded adversary $\mathcal{A}$ that makes at most a polynomial number of queries to $\mathcal{O}$, the advantage $\mathsf{Adv}_{\mathit{\Pi}_\Sigma}^{\mathsf{HVZK}}$ of $\mathcal{A}$ is negligible, where*

$$\mathsf{Adv}_{\mathit{\Pi}_\Sigma}^{\mathsf{HVZK}}(\mathcal{A}) := \left|\Pr[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, \mathsf{chall})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, \mathsf{chall})) = 1]\right|,$$

where $P = (P_1, P_2)$ is a prover running on $(\mathsf{st}, \mathsf{wt})$ with a challenge fixed to chall *and the probability is taken over the randomness used by (P, V) and by the random oracle.*

Definition 12 (2-Special Soundness). *We say a sigma protocol $\mathit{\Pi}_\Sigma$ has* 2-special soundness *if there exists a polynomial-time extraction algorithm such that, given a statement* st *and any two valid transcripts* $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *and* $(\mathsf{com}, \mathsf{chall}', \mathsf{resp}')$ *relative to* st *and such that* $\mathsf{chall} \neq \mathsf{chall}'$*, outputs a witness* wt *satisfying* $(\mathsf{st}, \mathsf{wt}) \in R$*.*

Remark 1. In some circumstances, we can relax the relation for 2-special soundness to R' where $R \subseteq R'$. That is, we allow the extractor to output wt such that $(\mathsf{st}, \mathsf{wt}) \in R'$ but $(\mathsf{st}, \mathsf{wt}) \notin R$. As long as given st to find wt such that $(\mathsf{st}, \mathsf{wt}) \in R'$, the sigma protocol can still serve as a proof system for some applications. For instance, [6,5] both relaxes the relation by including the collision of the hash function. Therefore, the resulting proof system, via the Fiat-Shamir transform, is sound if the hash function is collision-resistant.

Finally, note that sigma protocols can be used to build non-interactive zero-knowledge proofs of knowledge.

2.4 Proof Systems

We consider *non-interactive zero-knowledge proof of knowledge* (NIZKPoK) over the programmable random oracle model with statistical zero-knowledge and simulation-extractability.

Formally, a NIZKPoK for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\mathit{\Pi} = (\mathsf{Prove}, \mathsf{Verify})$ with access to the random oracle $\mathcal{O}$ such that for any valid public parameter of the scheme pp and the security parameter $\lambda \in \mathbb{N}$, the scheme satisfies the following properties:

- Completeness: for every $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}$,
$$\Pr[V^{\mathcal{O}}(\mathsf{st}, \pi) = 1 \mid \pi \leftarrow P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})] = 1.$$
- Statistical zero-knowledge: there exists a simulator Sim such that for any $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}$ and any possibly unbounded adversary $\mathcal{A}$ can distinguish the distributions: $\{\pi | \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})\}$, $\{\pi | \pi \leftarrow \mathsf{Sim}^{\mathcal{O}}(\mathsf{st})\}$ with only negligible advantage where the probability is taken over the randomness used in the experiment and the scheme.
- Simulation-extractability: for any adversary $\mathcal{A}$ there exists an extractor $\mathcal{E}_{\mathcal{A}}$ such that
$$\Pr\left[\begin{array}{r|r} \mathsf{Verify}(\mathsf{st}, \mathsf{wt}, \pi) = \top & (\mathsf{st}, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}, \mathcal{O}}(\mathsf{pp}) \\ (\mathsf{st}, \mathsf{wt}) \notin \mathcal{R} \ \wedge \ (\mathsf{st}, \mathsf{wt}, \pi) \notin L & \mathsf{wt} \leftarrow \mathcal{E}(\mathsf{st}, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$
where $\mathsf{SimProve}(\mathsf{st}', \mathsf{wt}')$ is an oracle that returns a simulated proof using $\pi' \leftarrow \mathsf{Sim}(\mathsf{st}')$ when $(\mathsf{st}', \mathsf{wt}') \in \mathcal{R}$. Otherwise, $\mathsf{SimProve}$ returns $\perp$. Also, $\mathsf{SimProve}$ maintains the list L, which is initially empty, and records $(\mathsf{st}', \mathsf{wt}', \pi')$ for each valid query of $(\mathsf{st}', \mathsf{wt}')$.

3 Malleable commitments

In this section we introduce the notion of malleable commitments. We will use this notion to build commitments based on group actions and isogenies from which we can derive zero-knowledge proofs for general NP statements. We first give a general definition together with the associated security notion and then restrict the notion to a specific kind of malleability which we will use later on in our constructions.

3.1 A generic notion of malleability

Intuitively, we say that a commitment scheme is *malleable* if given a commitment for an unknown value m, anyone can create a second commitment for some m' related to m. We formalise this notion below.

Definition 13 (Malleable Commitment). *Given a set $\mathcal{M}$ on which we have a relation R, a commitment scheme $(\mathcal{M}, \mathcal{R}, \mathcal{C}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ is said to be R-malleable if there exists a PPT algorithm $\mathcal{A}_m^R : \mathcal{C} \to \mathcal{C}$ with the following property: for any $\mathsf{com}' = \mathcal{A}_m^R(\mathsf{com})$, if there exists some $(m, r) \in \mathcal{M} \times \mathcal{R}$ such that $\mathsf{com} = \mathsf{Commit}(m, r)$, then there exists $(m', r') \in \mathcal{M} \times \mathcal{R}$ such that $\mathsf{com}' = \mathsf{Commit}(m', r')$ and $R(m, m')$.*

Remark 2. Depending on the application, r and r' might be equal. Additionally, the algorithm $\mathcal{A}_m$ does not have to be able to retrieve any of the values m, m' or r. However, in our constructions we only consider schemes for which whenever one party knows (m, r), they can retrieve the message m' corresponding to the commitment com'.

Remark 3. One can easily check that homomorphic commitments are a special case of malleable commitments for which $\mathcal{M}, \mathcal{R}$ and $\mathcal{C}$ are all groups.

As is, malleable commitments *a priori* do not impose any structure on $\mathcal{M}, \mathcal{R}$ or $\mathcal{C}$ and provide a very generic framework. Within this framework, we instantiate a specific type of malleable commitments exploiting the structure provided by isogeny-based cryptography. Namely, we enforce a group structure on the message and randomness spaces $\mathcal{M}$ and $\mathcal{R}$, while the commitment space $\mathcal{C}$ is only assumed to be a generic set. This is motivated by the fact that messages and randomness will be encoded as isogenies, which form a group under composition, while commitments will be seen as (isomorphism classes of) elliptic curves, which possesses no built-in group structure. The malleability will then be obtained by defining a group action of $\mathcal{M} \times \mathcal{R}$ on $\mathcal{C}$. In particular, since $\mathcal{C}$ is not a group, what we obtain is *not* an homomorphic commitment.
We formalize this additional structure required through the notion of *Admissible Group-Action-based Malleable Commitment* (AGAMC).

Definition 14 (Admissible group-action-based malleable commitment (AGAMC)). *A commitment scheme $\mathit{\Pi}_{\mathsf{Commit}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ is said to be an* admissible group-action based malleable commitment scheme *if the following requirements are satisfied:*

1. *The public parameters instantiated by* Setup *are* $\mathsf{pp} = (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$ *where* $\mathcal{M}$ *and* $\mathcal{R}$ *are (additive) abelian groups — with respective identities* $0_{\mathcal{M}}$ *and* $0_{\mathcal{R}}$*, and where* $X \in \mathcal{C}$ *is a distinguished element defined as* $X := \mathsf{Commit}(0_{\mathcal{M}}, 0_{\mathcal{R}})$.
2. *The action* $\star : (\mathcal{M} \times \mathcal{R}) \times \mathcal{C} \to \mathcal{C}$ *is regular and can be computed in polynomial-time. Any commitment is computed as* $\mathsf{Commit}(m, r) := (m, r) \star X$.
3. *The group action gives rise to the following malleability: for any* $(m, r), (m', r') \in \mathcal{M} \times \mathcal{R}$*, we have* $(m', r') \star \mathsf{Commit}(m, r) = \mathsf{Commit}(m + m', r + r')$.
4. *There are efficient uniform sampling methods for any subgroups of* $\mathcal{M}$*,* $\mathcal{R}$.
5. *There is an efficient algorithm to determine the membership of* $\mathcal{C}$.
6. *Every element in* $\mathcal{M}, \mathcal{R}, \mathcal{C}$ *has a unique representation.*

Remark 4. We do not require the algorithm Commit specified by pp to be computed efficiently itself. Instead, one uses malleability to produce a commitment $\mathsf{Commit}(m, r) = (m, r) \star \mathsf{Commit}(0_{\mathcal{M}}, 0_{\mathcal{R}})$. We abuse the notation 0 to represent $0_{\mathcal{M}}, 0_{\mathcal{R}}$. Also, we simplify the commitment scheme notion to be $\mathit{\Pi}_{\mathsf{Commit}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ and pp is implicitly used when the context is clear.

4 Malleable commitments from group actions

We present a construction of an AGAMC scheme from a known-order effective group action (Def. 5), and prove its security.
Construction. $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ proceeds as follows.

- $\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}$: on input λ, output $\mathsf{pp} = (G, G, \mathcal{E} \times \mathcal{E}, (E_0, E), \star')$ i.e. $\mathcal{M} = \mathcal{R} = G$, $\mathcal{C} = \mathcal{E} \times \mathcal{E}$ and $X := \mathsf{Commit}(0_{\mathcal{M}}, 0_{\mathcal{R}}) = (E_0, E)$, where $E = s \star E_0$ for some $s \in G$ and $s \neq 0$. The action $\star'$ is then defined as:

$$(m, r) \star' (Y_0, Y) := (r \star Y_0, (r + m) \star Y),$$

for $(G, \mathcal{E}, E_0, \star)$ a known-order effective group action where $N = |G|$ is given. The group size is implicitly parameterized by the security parameter λ.

- $\mathsf{Commit}(m, r) \to \mathsf{com}$: on input $(m, r) \in G^2$, returns $\mathsf{com} = (m, r) \star' X$.
- $\mathsf{Verify}(\mathsf{com}, (m, r)) \to 0/1$: on input $(\mathsf{com}, (m, r)) \in \mathcal{C} \times G^2$, checks whether $(m, r) \star' X = \mathsf{com}$. If yes, then outputs 1 and outputs 0 otherwise.

The requirements of an AGAMC follow naturally from the known-order effective group action. The uniqueness and membership test of $\mathcal{C}$ and the feasibility of $\star$ come from Def. 5. The uniform sampling methods of any subgroups of $\mathcal{M}, \mathcal{R}$ come from the known order and minimal generating set of G. Since we know minimal generating sets, we have a unique representation of $\mathcal{M}, \mathcal{R}$.

Theorem 1. *The commitment scheme described above is (computationally) hiding assuming DDH problem over $(G, \mathcal{E}, E_0, \star)$ is hard.*

Proof. Consider a DDH setup $(G, \mathcal{E}, E_0, \star)$. Given a DDH challenge (E_1, E_2, E_3) and access to an hiding adversary $\mathcal{A}$, we build an adversary $\mathcal{B}$ against DDH as follows.

1. Set $X = (E_0, E_1)$ and $\mathsf{pp} = (G, G, \mathcal{E} \times \mathcal{E}, X, \star')$. This gives a valid setup for our commitment construction.
2. Invoke the adversary $\mathcal{A}$ on the the public parameter pp.
3. Upon receiving $m_0, m_1 \in \mathcal{M}$ from $\mathcal{A}$, reply with $(E_2, m_b \star E_3)$ where $b \leftarrow \{0, 1\}$.
4. Output $(b = b')$ where b' is returned by $\mathcal{A}$.

We claim that this adversary $\mathcal{B}$ has half of the advantage of $\mathcal{A}$. Write $E_1 = s_1 \star E_0$ and $E_2 = s_2 \star E_0$ where s_1, s_2 are sampled uniformly from G by the DDH challenger. In the case that $E_3 = (s_1 + s_2) \star E_0$, i.e. $b = 1$, it is clear that $\mathcal{B}$ returns a well-formed commitment of m_b with respect to pp.

In the case that $E_3 \leftarrow \mathcal{E}$ by the challenger, the commitment is potentially malformed. Since the input $(E_2, m_i \star E_3)$ follows a uniform distribution over $\mathcal{E}^2$ regardless of m_b and b in the view of $\mathcal{A}$, the random bit b equals b' returned by $\mathcal{A}$ with probability exactly 1/2 in this case. Hence, the reduction $\mathcal{B}$ can win with probability exactly 1/2 in this case. Therefore,

$$\begin{aligned} |\Pr[\mathcal{B} \text{ wins}] - 1/2| &= |1/2 \Pr[\mathcal{B} \text{ wins} \mid E_3 = (s_1 + s_2) \star E_0] \\ &\quad + 1/2 \Pr[\mathcal{B} \text{ wins} \mid E_3 \neq (s_1 + s_2) \star E_0] - 1/2| \\ &= 1/2 |\Pr[\mathcal{B} \text{ wins} \mid E_3 = (s_1 + s_2) \star E_0] - 1/2| \\ &= 1/2 \ \mathsf{Adv}_{\Pi}^{\mathsf{Hide}}(\mathcal{A}). \end{aligned}$$

Hence, we have $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}) = \frac{1}{2} \mathsf{Adv}_{\Pi}^{\mathsf{Hide}}(\mathcal{A})$.

Theorem 2. *The commitment scheme described above is (perfectly) binding.*

Proof. Given $(m_0, r_0) \star' X = (m_1, r_1) \star' X$, we have $(r_0 \star Y_0, (r_0 + m_0) \star Y) = (r_1 \star Y_0, (r_1 + m_1) \star Y)$ for some $Y_0, Y \in \mathcal{E}$. Since the action is regular, we have $r_0 = r_1$ and $m_0 = m_1$, and the result follows.

4.1 Commitment products

We show that given two AGAMCs we can derive a new one via a direct product. Concretely, given two AGAMCs $\Pi_1 = (\mathsf{Setup}_1, \mathsf{Commit}_1, \mathsf{Verify}_1)$ and $\Pi_2 = (\mathsf{Setup}_2, \mathsf{Commit}_2, \mathsf{Verify}_2)$ we can define $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ in the following way: Given $\mathsf{Setup}_1 \to (\mathcal{M}_1, \mathcal{R}_1, \mathcal{C}_1, X_1, \star_1)$, $\mathsf{Setup}_2 \to (\mathcal{M}_2, \mathcal{R}_2, \mathcal{C}_2, X_2, \star_2)$ then $\mathsf{Setup} \to (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$ with $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$, $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$, $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$, $X = (X_1, X_2)$ and $((m_1, m_2), (r_1, r_2)) \star X = ((m_1, r_1) \star_1 X_1, (m_2, r_2) \star_2 X_2)$. The definitions of Commit and Verify follow accordingly.

Proposition 1 (Commitment Product). *Given two AGAMCs Π_1 and Π_2, then their product Π defined above gives an AGAMC. Moreover, Π is perfectly (resp. computationally) hiding (resp.binding) when both Π_1, Π_2 are perfectly (resp. computationally) hiding (resp.binding).*

Proof. The uniqueness and membership test of $\mathcal{M}, \mathcal{R}, \mathcal{C}$, the feasibility of $\star$, and the uniform sampling immediately follow from the properties of Π_1 and Π_2. Regarding the hiding property, by using a standard hybrid argument, for any hiding adversary $\mathcal{A}$ against Π, there exists B_1 against Π_1 and B_2 against Π_2 such that $\mathsf{Adv}^{\mathsf{Hide}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{Hide}}_{\Pi_1}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Hide}}_{\Pi_2}(\mathcal{B}_2)$. Regarding the binding property, suppose $(m, r) \star X = (m', r') \star X$ where $m = (m_1, m_2), m' = (m'_1, m'_2), r = (r_1, r_2), r' = (r'_1, r'_2)$. Then, we have $(m_1, r_1) \star_1 X_1 = (m'_1, r'_1) \star_1 X_1$ and $(m_2, r_2) \star_2 X_2 = (m'_2, r'_2) \star_2 X_2$. Therefore, Π is binding when both Π_1, Π_2 are binding.

Remark 5. Prop. 1 implies that the AGAMC is closed under the product. Alternatively, given an AGAMC with message and randomness space $\mathcal{M}$ and $\mathcal{R}$, we can extend both to vector spaces or matrix spaces $\mathcal{M}^{m \times n}, \mathcal{R}^{m \times n}$. As we will show in 6.2, this algebraic structure is sufficient to efficiently encode the information of circuits.

5 Proof systems for an admissible group-action based commitment

In this section, we build two proof systems for AGAMC that will serve as essential tools for constructing a generic proof system from this special commitment scheme. Both of them are proving the knowledge of the input of a commitment with a restricted message space. The nuance between the two protocols lies in the structure of the message space being a subgroup or not.
The first system (Sec. 5.1 and Fig. 1) will be needed to commit on the multiplicative gates of an arithmetic circuit (as $\mathcal{M}$ is not a *multiplicative* group) while the second (Sec. 5.2 and Fig. 2) will be used for the addition gates (and thus we can exploit the additive group structure of $\mathcal{M}$). The additional structure available in the second makes it more efficient : it is therefore simpler to commit on additions than on multiplications.

Formally, let $\mathsf{pp} = (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$, the parameters of an AGAMC and M' a subset of $\mathcal{M}$. Consider the following relation $\mathcal{R}_{\mathsf{pp}, M'} \subseteq \mathcal{C} \times \mathcal{M} \times \mathcal{R}$ where

$$\mathcal{R}_{\mathsf{pp},M'} = \{(\mathsf{st} = c, \mathsf{wt} = (m,r)) \mid c = (m,r) \star X \wedge m \in M'\}. \tag{1}$$

Equivalently, $\mathcal{R}_{\mathsf{pp},M'} = \{(\mathsf{st} = c, \mathsf{wt} = (m,r)) \mid c = \mathsf{Commit}(m,r) \wedge m \in M'\}$ due to the malleability. Therefore, to have a proof system for the relation $\mathcal{R}_{\mathsf{pp},M'}$ implies to have a proof system for the commitment without revealing m or r.

We then turn it into non-interactive zero-knowledge proof (NIZK) using parallel repetition and the Fiat-Shamir transform ([15]). Looking ahead, in Secs. 6.1 and 6.2 we will use this tool to build NIZK to prove satisfiability of arithmetic circuits and for rank-1 constraint systems (R1CS).

As mentioned earlier, commitment schemes from one-way functions can indeed be used to build generic zero-knowledge proof, but they result in highly inefficient ones. Homomorphic commitments (e.g. Pedersen commitments) have additional structure that makes them useful in many pre-quantum succinct zero-knowledge proofs, such as [10,26,25]. Here, we demonstrate that a malleable commitment (AGAMC) can also provide a structure suitable for use in post-quantum generic proof systems for arithmetic circuits when the underlying ring is small.

5.1 Proof system for small message space

We start with a fundamental proof system of AGAMC for proving the knowledge of the input of a commitment with an arbitrary small message space.

We first consider the case where M' is polynomially small. A concrete construction is given in Fig. 1 using Merkle trees to make the proof size logarithmic in $|M'|$. The use of Merkle trees is standard except for one modification: two leaves are concatenated in alphabetical order, resulting in an index-hiding Merkle tree [6]. To be more precise, the modification to the alphabetical order serves as a random (in the ROM) permutation of the input, ensuring that revealing a path in the tree does not reveal the location of the leaf. The construction is inspired by [5]. The high-level idea is that the prover uses the malleability property to create multiple new commitments $(-m', r')$ for each $m' \in M'$, and generates a Merkle root. The reason M' is required to be small is to make this possible. The prover is then challenged to either reveal all randomnesses used in the malleability, including the randomness used to generate the whole tree, or to reveal $(0, r + r')$ and the path of the tree that results in the same root. Intuitively, knowing the whole tree and the secret path allows for the determination of m along with r. Zero-knowledge follows from the uniform distribution of $r + r'$ (remember $\mathcal{R}$ has a group structure), and the use of an index-hiding Merkle tree to protect m.

Theorem 3. *The proof system described in Figure 1 is correct, 2-special sound (assuming collision resistance of the hash function H), and statistical honest-verifier zero-knowledge (HVZK) (in the random oracle model).*

Correctness. For the challenge $\mathsf{chall} = 0$, the correctness holds naturally. For the challenge $\mathsf{chall} = 1$, for the response $\mathsf{resp} = (r'', \mathsf{path}, \mathsf{bits})$, we have $r'' = r' + r$. Therefore, $\widetilde{\mathsf{com}} = (0, r'') \star X = (0, r' + r) \star X$. Recall that $\mathsf{com}_I = (-m_I, r') \star u$.

```
round 1: $P_1^{\mathcal{O}}((\mathsf{pp}, M', u), (m = m_I, r))$
seed $\leftarrow \{0,1\}^\lambda$
$(r', \mathsf{bits}_1, \cdots, \mathsf{bits}_t) \leftarrow H(\texttt{EXP}||\mathsf{seed})$        ▷ Sample $r' \in \mathcal{R}$ and $\mathsf{bits}_i \in \{0,1\}^\lambda$
for $i \in [t]$ do
    $\mathsf{com}_i \leftarrow (-m_i, r') \star u$
    $\mathsf{C}_i \leftarrow H(\texttt{COM}||\mathsf{com}_i||\mathsf{bits}_i)$        ▷ Create commitments $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N)$
$P$ sends comm $\leftarrow$ root to $V$.
```

```
round 2: $V_1(\mathsf{comm})$
$c \leftarrow \{0,1\}$
$V$ sends chall $\leftarrow c$ to $P$.
```

```
round 3: $P_2((m = m_I, r), \mathsf{chall})$
$c \leftarrow$ chall
if $c = 1$ then
    $r'' \leftarrow r' + r$
    path $\leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
    resp $\leftarrow (r'', \mathsf{path}, \mathsf{bits}_I)$
else
    resp $\leftarrow$ seed
$P$ sends resp to $V$.
```

```
Verification: $V_2^{\mathcal{O}}(\mathsf{comm}, \mathsf{chall}, \mathsf{resp})$
$(\mathsf{root}, c) \leftarrow (\mathsf{comm}, \mathsf{chall})$
if $c = 1$ then
    $(r'', \mathsf{path}, \mathsf{bits}) \leftarrow$ resp
    $\widetilde{\mathsf{com}} \leftarrow (0, r'') \star X$
    $\widetilde{\mathsf{C}} \leftarrow H(\texttt{COM}||\widetilde{\mathsf{com}}||\mathsf{bits})$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
else
    Repeat round 1 with seed $\leftarrow$ resp.
Output accept if the computation results in
root, and reject otherwise.
```

Figure 1. Our construction for the relation $\mathcal{R}_{\mathsf{pp},M'}$ (Eq. (1)) using an AGAMC with $\mathsf{pp} = (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$. The set M' is of cardinality t and can be publicly enumerated as $M' = \{m_1, \cdots, m_t\}$. The hash function H is modeled as a random oracle with prefixes EXP (for expand) and COM, for a pseudo-random number generator (PRNG) and a commitment scheme respectively and is also implicitly used in $\mathsf{MerkleTree}(\cdot)$.

Since $\mathsf{bits}_I = \mathsf{bits}$, we have $\widetilde{\mathsf{com}} = \mathsf{com}_I$ so that $\widetilde{\mathsf{C}} = \mathsf{C}_I$. $\mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$ will result in the same root and the correctness follows.

2-Special Soundness. Let $(\mathsf{comm} = \mathsf{root}, 0, \mathsf{resp}_0 = \mathsf{seed})$, $(\mathsf{comm} = \mathsf{root}, 1, \mathsf{resp}_1 = (r'', \mathsf{path}, \mathsf{bits}))$ be two valid transcripts. We can either extract a collision of the hash function H or a witness for the statement (pp, M', u). Following **round 1** of Fig. 1, say seed generates $r', \mathsf{com}_i = (-m_i, r') \star u$ for each $i \in [t]$. Let $\widetilde{\mathsf{com}} = (0, r'') \star X$. Since both $\mathsf{resp}_0, \mathsf{resp}_1$ result in the same root root, we assume $\widetilde{\mathsf{com}} = \mathsf{com}_{\widetilde{I}}$ for some $\widetilde{I} \in [t]$; otherwise, a collision of the hash function H can be found in the process of generating the root $(\mathsf{tree}, \mathsf{ReconstructRoot})$ due to the binding property of the Merkle tree [6, Lemma 2.9]. We claim that $(m_{\widetilde{I}}, r'' - r')$ is the witness. This is because we have $(0, r'') \star X = (-m_{\widetilde{I}}, r') \star u$. Therefore, $(m_{\widetilde{I}}, r'' - r') \star X = u$. The result follows.

Statistical HVZK. We show the scheme is statistical HVZK by modeling the hash function H as a random oracle, which is essential to achieve statistical zero-knowledge. For the challenge $\mathsf{chall} = 0$, the simulator follows the first round of Fig. 1 and generates $(\mathsf{seed}, 0, \mathsf{comm})$ as the transcript. For the challenge $\mathsf{chall} = 1$, the simulator generates r'' from $\mathcal{R}$ uniformly at random and computes $\widetilde{\mathsf{com}_1} = (0, r'') \star X$. Then, the simulator generates uniformly r' from $\mathcal{R}$ and $\mathsf{bits}_1, \cdots, \mathsf{bits}_t$ from $\{0,1\}^\lambda$, computes $\widetilde{\mathsf{C}_1}(\texttt{COM}||\widetilde{\mathsf{com}_1}||\mathsf{bits}_1)$ and $\mathsf{C}_i = H(\texttt{COM}||(-m_i, r') \star u||\mathsf{bits}_i)$ for $i \in \{2, \cdots, t\}$, and generates $(\mathsf{root}, \mathsf{path}) \leftarrow \mathsf{MerkleTree}(\widetilde{\mathsf{C}_1}, \mathsf{C}_2, \cdots, \mathsf{C}_N)$. Finally, the simulator outputs the transcript $(\mathsf{root}, 1, (r'', \mathsf{path}, \mathsf{bits}_1))$. Even though the simulator simulates for $I = 1$, the distribution is independent of this partic-

```
round 1: $P_1^{\mathcal{O}}((\mathsf{pp}, M', u), (m, r))$
$\mathsf{seed} \leftarrow \{0,1\}^{\lambda}$
$(m', r') \leftarrow H(\mathtt{EXP}||\mathsf{seed})$                ▷ Sample $(m', r') \in M' \times \mathcal{R}$
$\mathsf{C} \leftarrow H(\mathtt{COM}||(m', r') \star u)$          ▷ Create commitments $\mathsf{C} \in \{0,1\}^{2\lambda}$
$P$ sends $\mathsf{comm} \leftarrow \mathsf{C}$ to $V$.
```

```
round 2: $V_1(\mathsf{comm})$
$c \leftarrow \{0,1\}$
$V$ sends $\mathsf{chall} \leftarrow c$ to $P$.
```

```
round 3: $P_2((m, r), \mathsf{chall})$
$c \leftarrow \mathsf{chall}$
if $c = 1$ then
    $(m'', r'') \leftarrow (m' + m, r' + r)$
    $\mathsf{resp} \leftarrow (m'', r'')$
else
    $\mathsf{resp} \leftarrow \mathsf{seed}$
$P$ sends $\mathsf{resp}$ to $V$.
```

```
Verification: $V_2^{\mathcal{O}}(\mathsf{comm}, \mathsf{chall}, \mathsf{resp})$
$(\mathsf{C}, c) \leftarrow (\mathsf{comm}, \mathsf{chall})$
if $c = 1$ then
    $(m'', r'',) \leftarrow \mathsf{resp}$
    $\widetilde{\mathsf{C}} \leftarrow H(\mathtt{COM}||(m'', r'') \star X)$
else
    Repeat round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
Output $\mathsf{accept}$ if $\widetilde{\mathsf{C}} = \mathsf{C}$ or round 1 with
  $\mathsf{seed} \leftarrow \mathsf{resp}$ leads to $\mathsf{C}$
```

Figure 2. Our construction for the relation $\mathcal{R}_{\mathsf{pp},M'}$ (Eq. (1)) using an AGAMC with $\mathsf{pp} = (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$ where M' is a subgroup of $\mathcal{M}$. The hash function H is modeled as a random oracle with prefixes EXP and COM as a PRNG and a commitment scheme respectively.

ular choice due to the hiding property of the Merkle tree [6, Lemma 2.10]. The only difference is that $r', \mathsf{bits}_1, \cdots, \mathsf{bits}_t$ are not generated by $H(\mathtt{EXP}||\cdot)$. Since seed has λ bits of min-entropy and is information-theoretically hidden from the distinguisher, for any unbounded distinguisher with Q queries to H of the form $H(\mathtt{EXP}||\cdot)$ the statistical difference between the distributions is $Q_H/2^{\lambda}$.

Efficiency. Suppose it requires x bits to represent an element over the message space M' and y bits for the randomness space $\mathcal{R}$. The scheme in Fig. 1 has proof size $\frac{\lambda+(y+\log\lambda|M'|+\lambda)}{2}$ bits on average and requires $O(|M'|)$ malleability actions.

5.2 Proof system for message spaces with a subgroup structure

In Sec. 5.1, we require the message space to be small since we do not have any assumptions on its structure. Here we show that we can remove this restriction when the message space M' is a subgroup of $\mathcal{M}$, we obtain a more efficient NIZK based on the proof system shown in Fig. 2.

Theorem 4. *The proof system described in Fig. 2 is correct, 2-special sound (assuming collision resistance of the hash function H) and statistical honest verifier zero-knowledge.*

Proof. See Appendix C.1 in the full version of the paper.

Efficiency. Say it requires x bits to represent an element over the message space M' and y bits for the randomness space $\mathcal{R}$. The scheme in Fig. 2 has proof size $\frac{\lambda+(x+y)}{2}$ bits on average and requires 2 malleability actions.

5.3 NIZK via the Fiat-Shamir Transform

By applying the Fiat-Shamir transform [15], we obtain NIZKs for the relation $\mathcal{R}_{\mathsf{pp},M'}$ as shown in Fig. 3. Using Theorems 3 and 4 respectively, and the properties of the Fiat-Shamir transform we may prove that the NIZKs are complete and statistical zero-knowledge with standard proofs.

In order to demonstrate simulation-extractability, we can reuse the proof of Thm. 6.1 in [5] and show that our NIZKs are also (multi-proof) online-extractable (see Def. 2.10 of [5] for the definition), which implies the simulation extractability and has online-extractability. From a high level, the scheme is online-extractable because we use the PRNG to generate the randomness in the commitment in the sigma protocol. In the security proof, the PRNG is modeled as a random oracle and the challenge is binary. Alternatively, a simulator can extract the response for the challenge 0 by checking the random oracle queries regardless of the challenge $c \in \{0,1\}$. In other words, since we use the random oracle as PRNG to generate the randomness for the commitment and the challenge is binary, the proof of Theorem 6.1 in [5] is agnostic to the underlying sigma protocol by replacing Step 2.(b) with the 2-special soundness extractor. We, therefore, omit the proof for simplicity.

Theorem 5. *The NIZKs of Figs. 1 and 2 compiled by Fig. 3 is complete, simulation-extractable and statistically zero-knowledge in the random oracle model.*

```
$\mathsf{Prove}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})$
for $i \in [\lambda]$ do
    $\mathsf{com}_i \leftarrow P_1^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})$
$\mathsf{com} \leftarrow (\mathsf{com}_1, \cdots, \mathsf{com}_\lambda)$
$(c_1, \cdots, c_\lambda) \leftarrow \mathcal{O}(\mathsf{FS}||\mathsf{st}||\mathsf{com})$
$\mathsf{chall} \leftarrow (c_1, \cdots, c_\lambda)$
for $i \in [\lambda]$ do
    $\mathsf{resp}_i \leftarrow P_2^{\mathcal{O}}(\mathsf{st}, \mathsf{com}_i, c_i)$
$\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$
return $\pi \leftarrow (\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
```

```
$\mathsf{Verify}^{\mathcal{O}}(\mathsf{st}, \pi)$
$(\mathsf{com} = (\mathsf{com}_1, \cdots, \mathsf{com}_\lambda), \mathsf{chall} = (c_1, \cdots, c_\lambda), \mathsf{resp} = (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)) \leftarrow \pi$
$\mathsf{output} = 1$
for $i \in [\lambda]$ do
    $r \leftarrow V_2(\mathsf{com}_i, c_i, \mathsf{resp}_i)$
    $\mathsf{output} \leftarrow \mathsf{output} \cdot r$
$\mathsf{output} \leftarrow \mathsf{output} \cdot (\mathsf{chall} == \mathcal{O}(\mathsf{FS}||\mathsf{st}||\mathsf{com}))$
return output
```

Figure 3. NIZK for the relation by applying the Fiat-Shamir transform to $\mathit{\Pi} = (P = (P_1, P_2), V = (V_1, V_2))$ from either Fig. 1 or Fig. 2 with λ repetitions.

6 Proof Systems for NP Statements

This section presents proof systems for an arithmetic circuit over a finite field or ring constructed from an AGAMC. Firstly, we consider two cases: the arithmetic circuit in general with the addition and multiplication gates in Sec. 6.1, and the representation using rank-1 constraint system in Sec. 6.2. In both cases, the underlying finite field or ring must be small or embedded into a small subgroup of the message space. In Sec. 6.3, we construct a proof system for the branching program representation without size restrictions.

6.1 Arithmetic circuits over a small ring

This subsection gives a NIZK for the satisfiability of an arithmetic circuit over a small ring. We encode this ring as $M' \subseteq \mathcal{M}$ of a given AGAMC $\mathit{\Pi} = (\mathcal{M}, \mathcal{R}, \mathcal{C}, X, \star)$ (e.g. via a dictionary map).

The idea is fairly straightforward. For each gate of the circuit, the prover uses an AGAMC to make a commitment to the inputs and the output. Then, the prover uses Fig. 1 to generate a proof based on the commitment product Prop. 1.Concretely, for the input (m_1, m_2) and the output m_3 for a certain gate. The prover computes $\mathsf{Commit}(m_1, r_1)$, $\mathsf{Commit}(m_2, r_2)$, and $\mathsf{Commit}(m_3, r_3)$. Say the ring of the circuit is encoded as $M' \subseteq \mathcal{M}$. The prover uses Fig. 1 for the triple commitment $\mathit{\Pi} \times \mathit{\Pi} \times \mathit{\Pi}$ with the message space $M'' \subseteq M'^3$ by collating the valid input-output tuple corresponding to the gate. For example, consider an addition gate with inputs m_1, m_2 and output m_3. We need to prove that $\mathsf{Commit}(m_1, r_1)$, $\mathsf{Commit}(m_2, r_2)$, and $\mathsf{Commit}(m_3, r_3)$ are all in $\mathit{\Pi}$ and that $m_1 + m_2 = m_3$. Regarding a multiplication gate with inputs m_1, m_2 and output m_3. We need to prove that $\mathsf{Commit}(m_1, r_1)$, $\mathsf{Commit}(m_2, r_2)$, and $\mathsf{Commit}(m_3, r_3)$ are all in $\mathit{\Pi}$ and that $m_1 * m_2 = m_3$.

Efficiency. Say it requires x bits to represent an element over the message space M' and y bits for the randomness space $\mathcal{R}$. Let the number of parallel repetitions of the base schemes Figs. 1 and 2 be λ in Fig. 3. The scheme in Fig. 1 has proof size $\frac{\lambda^2 + \lambda(3y + x \log \lambda + 3\lambda)}{2} x$ bits on average and requires $O(|M'|^3)$ malleability actions. The equality test can also be done in the same way. Say the prover commits to $\mathsf{Commit}(m_1, r_1)$, $\mathsf{Commit}(m_1, r_2)$. Then, the prover uses Fig. 1 for the commitment product $\mathit{\Pi} \times \mathit{\Pi}$ with the message space $M'' \subseteq M'^2$ by collecting $(m, m) \in M'^2$.

Optimization for a group M'. When the ring of the circuit can be embedded as a subgroup $M' \leq \mathcal{M}$, the proofs for a variety of gates can be significantly improved, both with respect to proof size and efficiency.

Concretely, we can improve the proofs of the gates of equalities, additions, addition-by-a-constant, or multiply-by-a-constant. This is because the tuple of valid input-output tuples will form a subgroup of $\mathcal{M}^3$ by Prop. 2. In this case we can use Fig. 2 to generate the proofs. In contrast, the proof size is improved by a factor of $\log_2(M'^2)$ and the number of actions is reduced by a factor of M'.

However, the main bottleneck of our technique is rooted in the multiplication gate where the input-output tuples cannot form a subgroup over $\mathcal{M}^3$. Therefore, it still requires Fig. 1 to generate a proof for a multiplication gate and the proof size is of $O(\log(M'))$. This is also the reason that our result cannot be extended to a larger ring in a computationally feasible manner.

6.2 Proof System for Rank-1 Constraint System Over a Small Ring

Let $\mathbb{F}$ be a finite field (or a ring), the rank-1 Constraint System (R1CS) relation consists of statement-witness pairs $((A, B, C, v), w)$ where $A, B, C \in \mathbb{F}^{m \times (n+1)}$

and v, w are matrices such that $(Az) \circ (Bz) = Cz$ for $z := (1, v, w) \in \mathbb{F}^{n+1}$ and $\circ$ denotes the entry-wise product (i.e. the Hadamard product). R1CS capture arithmetic circuit satisfaction : A, B, C encode the circuit's gates, the circuit's public input v, and w is the circuit's private input and wire values.

Recall from Rem. 5 that we can assume the message space of an AGAMC to be a vector. Therefore, by using an AGAMC, a prover generates proofs for the rowcheck and lincheck problems as follows.

1. Commit to $z := (1, v, w), Az, Bz, Cz$ using Commit. Let $\mathsf{C}_z, \mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C$ be the resulting commitment vectors.
2. **Rowcheck:** Generate the proof for $\mathsf{C}_z, \mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C$ using Fig. 2 for the relation
$$\left\{ \begin{array}{l|l} & \mathsf{C}_z = \mathsf{Commit}(z; r_z), \\ (\ \mathsf{st} = (A, B, C, \mathsf{C}_z, \mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C), & \mathsf{C}_A = \mathsf{Commit}(Az; r_A), \\ \quad \mathsf{wt} = (z, r_z, r_A, r_B, r_C)\ ) & \mathsf{C}_B = \mathsf{Commit}(Bz; r_B), \\ & \mathsf{C}_C = \mathsf{Commit}(Cz; r_C). \end{array} \right\}.$$
3. **Lincheck:** Generate the proof for $\mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C$ using Fig. 1 for the relation
$$\left\{ \begin{array}{l|l} & \mathsf{C}_A = \mathsf{Commit}(y_A; r_A), \\ (\ \mathsf{st} = (\mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C), & \mathsf{C}_B = \mathsf{Commit}(y_B; r_B), \\ \quad \mathsf{wt} = (y_A, y_B, y_C, r_A, r_B, r_C)\ ) & \mathsf{C}_C = \mathsf{Commit}(y_C; r_C), \\ & y_C = y_A \circ y_B. \end{array} \right\}.$$
4. Output $\mathsf{C}_z, \mathsf{C}_A, \mathsf{C}_B, \mathsf{C}_C$ and the proofs.

Note that we are allowed to apply Fig. 2 to the rowcheck because if M' is a subgroup of $\mathcal{M}$, where $\mathcal{M}$ is of dimensional $n+1$, then $\{(m, Am, Bm, Cm) | m \in M'\}$ is also a subgroup of $\mathcal{M}^4$, which can be formalized as Prop. 2. In contrast, Fig. 2 is not applicable to the lincheck due to the non-subgroup structure of the restriction $y_C = y_A \circ y_B$.

Proposition 2. *Let M_1, M_2 be subgroups of a finite abelian group $(\mathcal{M}, +)$. Then,*

(i) $M_1 \times M_2$ is also a subgroup of $\mathcal{M}^2 := \mathcal{M} \times \mathcal{M}$.

(ii) the set $\{(x, x) \subseteq M_1^2\}$ is a subgroup of $\mathcal{M}^2$.

(iii) for any integer $a, b, c \in \mathbb{Z}$, the set $\{(x, y, z) \subseteq M_1 \times M_2 \times \langle M_1, M_2 \rangle \mid ax + by = cz\}$ is a subgroup of $\mathcal{M}^3$.

For a given instantiation of an AGAMC scheme, the estimation is given in Tab. 1 with $A, B, C \in \mathbb{F}^{m \times (n+1)}$ and $z \in \mathbb{F}^{1+n}$ for an arithmetic circuit over a finite field $\mathbb{F}$ which can be embedded as a subgroup into $\mathcal{M}$. For example, when $\lambda = 128$ with an instantiation of AGAMC using CSIDH-2048 (assuming the existence), we have $\eta \approx \gamma^2 \approx 2048$.

Item	Cost
pp	$\frac{\eta}{4}$ B
Commitment	$(3m+n+1)\frac{\eta}{8}$ B
Rowcheck Proof	$\frac{\lambda\rho}{16}(\lambda+(3m+n+1)\log_2(\lvert\mathbb{F}\rvert)+\gamma(3m+n+1)$ B
Lincheck Proof	$\frac{\lambda\rho}{16}(\lambda+\eta(m+\lambda\log_2(\lvert\mathbb{F}\rvert)))$ B
Computational Cost	$\mathcal{O}(mn\lambda\lvert\mathbb{F}^2\rvert)$

Table 1. The proof (both rowcheck and lincheck proofs) size estimation of R1CS over $\mathbb{F}$ where $\mathbb{F}$ can be embedded as a subgroup using Figs. 1 and 2. The parameters η, γ are the numbers of bits to represent the elements of $\mathcal{C}$ and $\max\{|\mathcal{M}|, |\mathcal{R}|\}$, respectively. The density $\rho \in (0,1)$ is the Hamming weight of A, B, C divided by $3m(n+1)$. The computational cost is evaluated for both parties in terms of the malleability operations.

6.3 Zero-knowledge proofs for branching programs

Barrington's theorem is a fundamental result in computational complexity theory that has important implications for circuit design. It states that any circuit of depth d and fan-in 2 can be represented by a matrix branching program of length at most 4^d and width 5 [3].

In more formal terms, a matrix branching program over a field $\mathbb{F}$ of depth d and width w for ℓ-bit inputs is defined by a sequence $\mathsf{BP} = (i, A_{i,0}, A_{i,1})_{i\in[d]}$, where $A_{i,b}$ is a square matrix of $\mathbb{F}$ of dimension w for $i \in [d]$ and $b \in \{0,1\}$. The evaluation function e maps each index $i \in [d]$ to a bit position $e(i) \in [\ell]$.

Given an input $x \in \{0,1\}^\ell$, the matrix branching program evaluates by computing the product of matrices $\Pi_{i=1}^{d} A_{i,x_{e(i)}}$. The program outputs 1 if and only if the resulting product equals the identity matrix.

In this section, we show how to prove the satisfiability of a circuit in the form of a branching program (over a ring) using AGAMCs. We start by introducing the high-level idea. Let N be a natural number and $(i, A_{j,0}, A_{j,1})_{j\in[d]}$ represent a matrix branching program defined over $\mathbb{Z}_N$, with depth d and width w and let square matrices $M_0 = I_w, (M_j)_{j\in[d]}$ be a transcript of the program where $M_1 = A_{1,0}M_0$ or $= A_{1,1}M_0$, depending on the input x and the evaluation function e, and M_d is the final product.

Our proof system of the matrix branching program will proceed as follows.

1. Commit to M_j via $u_j = \mathsf{Commit}(M_j, R_j)$.
2. Compute $M_{j+1} = A_{j,b}M_j$ depending on the secret $b \in \{0,1\}$.
3. Commit to M_{j+1} via $u_{j+1} = \mathsf{Commit}(M_{j+1}, R_{j+1})$.
4. Generate a proof that the values committed in u_{j+1} and u_j are related by either $A_{j,0}$ or $A_{j,1}$ (i.e. $M_{j+1} = A_{j,0}M_j$ or $A_{j,1}M_j$).

It suffices to build a proof system for Item 4, as described in the following.

Base proof system This subsection introduces a fundamental tool using AGAMC for our proof system of a circuit in the form of a branching program. Concretely, for a given parameter $\mathsf{pp} = (\mathcal{M}'^2, \mathcal{R}'^2, \mathcal{C}'^2, X, \star)$, where $\mathcal{M}', \mathcal{R}', \mathcal{C}'$ correspond to the message space, the randomness space and the commitment space in the square matrix variant as explained in Rem. 5, we consider the following relation

$$\mathcal{R}_{\mathsf{pp},A_0,A_1} = \left\{ \begin{array}{l|c} (\ \mathsf{st} = (u_1, u_2), & M_1, M_2 \in \mathcal{M}' \\ \ \ \mathsf{wt} = ((M_1, M_2), (R_1, R_2))) & u_1 = (M_1, R_1) \star X \ \wedge \\ \ \ \in \mathcal{C}'^2 \times \mathcal{M}'^2 \times \mathcal{R}'^2 & u_2 = (M_2, R_2) \star X \ \wedge \\ & A_0 M_1 = M_2 \ \vee A_1 M_1 = \ M_2 \end{array} \right\}.$$

Note that a standard OR-proof [12] does not seem feasible here. An OR-proof requires statements for both relations and proves the knowledge of a witness for one of the two statements. Therefore, using OR-proof for a branching program will induce a statement and the proof size growing exponentially in the length because the prover needs to provide all possible statements in advances.

In contrast, we construct a proof system with linear growth as in Fig. 4. The high-level idea is that the prover generates and commits to (M', A_0M', R'),(M', A_1M', R'), then reveals either $(M_b{+}M', R{+}R')$ or (M', R'). However, this will disclose the information of b when the response is the $(M_b + M', R + R')$ and the witness is given. To this end, the prover additionally commits to $(M_1 + M', A_{1-b}(M_1 + M'), R + R')$ and shuffles with the previous commitments in a specific way to hide the information of b.

Notations. When the context is clear, we abuse notation by writing $(M_1', M_2', R_1', R_2') \star (x_1, x_2)$ to represent $((M_1', R_1') \star x_1, (M_2', R_2') \star x_2)$.

```
round 1: $P_1^{\mathcal{O}}((\mathsf{pp}, A_0, A_1, u = (u_1, u_2)), (M_1, M_2, R = (R_1, R_2), b))$
Parse $\mathsf{pp} = (\mathcal{M}'^2, \mathcal{R}'^2, \mathcal{C}'^2, X, \star)$
$\mathsf{seed}, \mathsf{bits}_0, \mathsf{bits}_1, \mathsf{bits}_2 \leftarrow\$ \{0,1\}^\lambda$
$(M', R') \leftarrow H(\mathtt{EXP}||\mathsf{seed})$    ▷ Sample $(M', R') \in \mathcal{M}' \times \mathcal{R}'^2$ and $\mathsf{bits}_i \in \{0,1\}^\lambda$
$\mathsf{C}_b \leftarrow H(\mathtt{COM}||(M', A_0M', R') \star u||\mathsf{bits}_b)$    ▷ Commitments $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
$\mathsf{C}_{b+1} \leftarrow H(\mathtt{COM}||(M', A_1M', R') \star u||\mathsf{bits}_{b+1})$
$\mathsf{C}_{2-2b} \leftarrow H(\mathtt{COM}||(M_1 + M', A_{1-b}(M_1 + M'), R + R') \star (X, X)||\mathsf{bits}_{2-2b})$
$\mathsf{root} \leftarrow \mathsf{MerkleTree}((\mathsf{C}_0||\mathsf{C}_1||\mathsf{C}_2), (\mathsf{C}_2||\mathsf{C}_0||\mathsf{C}_1))$    ▷ Index-hiding Merkle Tree
Prover sends $\mathsf{comm} \leftarrow \mathsf{root}$ to Verifier.

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

round 3: $P_2((M, R, b), \mathsf{chall})$
$c \leftarrow \mathsf{chall}$
if $c = 1$ then
    $(M'', R'') \leftarrow (M_1 + M', R + R')$
    $\mathsf{resp} \leftarrow (M'', R'', \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1)$
else
    $\mathsf{resp} \leftarrow (\mathsf{seed}, \mathsf{bits}_b, \mathsf{bits}_{b+1}, \mathsf{C}_{2-2b})$
Prover sends $\mathsf{resp}$ to Verifier

Verification: $V_2^{\mathcal{O}}(\mathsf{comm}, \mathsf{chall}, \mathsf{resp})$
$(\mathsf{root}, c) \leftarrow (\mathsf{comm}, \mathsf{chall})$
if $c = 1$ then
    $(M'', R'', \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1) \leftarrow \mathsf{resp}$
    $\widetilde{\mathsf{C}} \leftarrow H(\mathtt{COM}||(M'', A_0M'', R'') \star (X, X)||\mathsf{bits}_0)$
    $\widetilde{\mathsf{C}}' \leftarrow H(\mathtt{COM}||(M'', A_1M'', R'') \star (X, X)||\mathsf{bits}_2)$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{MerkleTree}((\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}'), (\widetilde{\mathsf{C}}'||\widetilde{\mathsf{C}}||\mathsf{C}_1))$
else
    $(\mathsf{seed}, \mathsf{bits}_b, \mathsf{bits}_{b+1}, \mathsf{C}_{2-2b}) \leftarrow \mathsf{resp}$
    $(M', R') \leftarrow H(\mathtt{EXP}||\mathsf{seed})$
    $\widetilde{\mathsf{C}} \leftarrow H(\mathtt{COM}||(M', A_0M', R') \star u||\mathsf{bits}_b)$
    $\widetilde{\mathsf{C}}' \leftarrow H(\mathtt{COM}||(M', A_1M', R') \star u||\mathsf{bits}_{b+1})$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{MerkleTree}((\widetilde{\mathsf{C}}||\widetilde{\mathsf{C}}'||\mathsf{C}_{2-2b}), (\mathsf{C}_{2-2b}||\widetilde{\mathsf{C}}||\widetilde{\mathsf{C}}'))$
Output $\mathsf{accept}$ if $\widetilde{\mathsf{root}} = \mathsf{root}$, and $\mathsf{reject}$ otherwise.
```

Figure 4. A three-move interactive protocol $(P^{\mathcal{O}} = (P_1^{\mathcal{O}}, P_2), V = (V_1, V_2^{\mathcal{O}}))$ to prove the possession of a witness (M_1, M_2, R_1, R_2, b) for a statement $u = (u_1, u_2)$ such that $u = (M_1, M_2, R_1, R_2) \star (X, X)$ and $M_2 = A_bM_1$.

Theorem 6. *In the random oracle model, by assuming* $\mathsf{MerkleTree}$ *to be collision-resistant and* $H(\mathtt{COM}||\cdot)$ *is binding, the construction* $(P^{\mathcal{O}} = (P_1^{\mathcal{O}}, P_2)$, $V = (V_1, V_2^{\mathcal{O}}))$ *in Fig. 4 has correctness, 2-special soundness, and HVZK for the relation* $\mathcal{R}_{\mathsf{pp},A_0,A_1}$ *with* Q_H *queries to* H.

Proof. The correctness holds clearly when the challenge bit is 0. When the challenge is 1 the verifier have $\mathsf{resp} = (M'', R'', \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1)$ where $M'' = M_1 + M'$ and $R'' = R + R'$. Then, we have either $\{\widetilde{\mathsf{C}}, \widetilde{\mathsf{C}}'\} = \{\mathsf{C}_0, \mathsf{C}_2\}$ where $\widetilde{\mathsf{C}} = H(\mathtt{COM}||(M'', A_0 M'', R'') \star (X, X)||\mathsf{bits}_0)$ and $\widetilde{\mathsf{C}}' \leftarrow H(\mathtt{COM}||(M'', A_1 M'', R'') \star (X, X)||\mathsf{bits}_2)$. Therefore, the Merkle tree on input $(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}'), (\widetilde{\mathsf{C}}'||\widetilde{\mathsf{C}}||\mathsf{C}_1)$ and $(\mathsf{C}_0||\mathsf{C}_1||\mathsf{C}_2), (\mathsf{C}_2||\mathsf{C}_0||\mathsf{C}_1)$ will result in the same root since the entries of the input will be ordered alphabetically. We leave the proof for 2-special soundness and honest-verifier zero-knowledge to Appendix C.2 in the full version of this paper.

Remark 6. The Merkle tree only has two leaves in **round 1** so the prover does not need to store additional information for the tree to generate the response.

Proof Size and Computational Cost. We turn Fig. 4 into a NIZK via Fig. 3, giving an essential tool for our proof system for a branching program.

Let N be a natural number and $(i, A_{j,0}, A_{j,1})_{j\in[d]}$ represent a matrix branching program defined over $\mathbb{Z}_N$, with depth d and width w. For a given instantiation of an AGAMC scheme, the estimation is given in Tab. 2 using as described in Rem. 5 to produce the commitment scheme in the matrix form. For example, regarding the proof size, when $\lambda = 128$ with an instantiation of AGAMC using CSIDH-2048 (assuming the existence), we have $\eta \approx \gamma^2 \approx 2048$.

Regarding the computational complexity, as shown in Tab. 2, we measure the cost in terms of the malleability operation, which is the group action operation that gives malleability. Within each layer $j \in [d]$, both the prover and verifier need to perform $O(\lambda)$ malleability operation. Therefore, the proof takes both parties $O(d\lambda w^2)$ CSIDH group actions regardless of the size of the underlying ring. This cost estimate applies to each layer of the proof and takes into account the size of the matrices used in the computation.

Item	Cost (Bytes)
pp	2η B
Commitment	$dw^2 \frac{\eta}{8}$ B
Proof	$\frac{\lambda d}{16}((N + 2\gamma)w^2 + 9\lambda)$ B
Computational Cost	$\mathcal{O}(\lambda d w^2)$

Table 2. An evaluation of the commitment size and the proof size Fig. 4 for a matrix branching problem over $\mathbb{Z}_N$ and of length d and width w, where N is also the number of bits to represent $\mathcal{M}$. The parameters η, γ are the numbers of bits to represent the elements of $\mathcal{C}$ and $\mathcal{R}$, respectively. Note that $(\mathcal{C}', \mathcal{M}', \mathcal{R}') = (\mathcal{C}^{w\times w}, \mathcal{M}^{w\times w}, \mathcal{R}^{w\times w})$. The computational cost is evaluated for both parties given in terms of the malleability operations.

6.4 Discussion and further work

Compared to the classical setting, there are three major challenges in our current context that require further attention.

The first bottleneck, concerning our proof systems for generic arithmetic circuits over a ring (Secs. 6.1 and 6.2) is that to prove the multiplication relation of the committed messages, we employ an OR-proof-type proof system. Although the proof remains logarithmic in the ring size (of the arithmetic circuit), both the prover and the verifier must compute the actions of all possible combinations of (M_1, M_2, M_1M_2). This limits the ring size as well as the message space, as they cannot both be large. Developing a more efficient proof system for the multiplication relation of the committed messages would improve the capability of the proof systems in Secs. 6.1 and 6.2. Note however, that this restriction *does not affect* the proof system for branching program, which relies on an affine relation, as discussed in Sec. 6.3.

Another restriction is the lack of an efficient commitment scheme that allows a prover to commit to a vector of messages using only one element. In classical cryptography, the generalized Pedersen commitment provides such a functionality. In our setting, a naive approach is to use randomly generated elements $y, g_1, \cdots, g_n$, and Y_0, Y from the group, and commit to $(y^r \star Y_0, (r \prod_i^n g_i^{m_i}) \star Y)$ for the messages $m_1, \cdots, m_n$ (using multiplicative notation for the group operation). However, this commitment is not quantumly binding since the relations between the g_i's can be recovered using a quantum period-finding algorithm. This is a challenge that is worth investigating in future research.

Finally, the folding technique, which enables proof compression [10,11] does not extend from homomorphic to malleable commitment or the Naor-Reingold-type VRF [22], resulting in an argument with square root communication complexity. A variant of that technique making use of the malleability rather than the homomorphic property is an interesting question to explore.

However, our construction does offer several advantages. First, we do not require any form of trusted set-up. Second, we achieve statistical, rather than computational, zero-knowledge, as well as simulation extractability, meaning that an adversary cannot simulate a new proof unless they know a witness, regardless of the number of simulated proofs they have seen. This is a stronger notion than soundness or even knowledge soundness, and highly desirable in situations where the non-malleability of the proofs is required. Finally, we obtain linear efficiency in the size of circuit. Despite being less efficient than state of the art post-quantum zero-knowledge proofs, such as [2,4,8] whose proofs sizes are sublinear or even polylogarithmic in the size of the witness, our approach still improves on the naive construction from one-way functions and provides a first general purpose zero-knowledge proof from isogenies.

7 Conclusion

In this paper, we introduce and formalize the notion of malleable commitment, which extends the notion of homomorphic commitments but enforces less structure on the underlying commitment scheme. We show how to instantiate such malleable commitments from an isogeny-based group action. Subsequently, we build different flavors of generic proofs for NP statements. Our first two approaches, based on arithmetic circuits and rank-1 constraint systems impose restrictions on the size of the underlying ring and message space. However, this restriction is lifted by our third approach based on branching programs. Regardless, these three approaches offer the first feasibility results towards post-quantum zero-knowledge proofs based on isogeny assumptions, improving on the naive construction from one-way functions.

Acknowledgements

Mingjie Chen and Christophe Petit are partly supported by EPSRC through grant number EP/V011324/1. Yi-Fu Lai thanks the New Zealand Ministry for Business and Employment for financial support.

References

1. N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. Cryptographic group actions and applications. *ASIACRYPT 2020, Part II*, pp. 411–439.
2. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. *ACM CCS 2017*, pp. 2087–2104.
3. D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 1–5.
4. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. *EUROCRYPT 2019, Part I*, pp. 103–128.
5. W. Beullens, S. Dobson, S. Katsumata, Y.-F. Lai, and F. Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. *EUROCRYPT 2022, Part II*, pp. 95–126.
6. W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. *ASIACRYPT 2020, Part II*, pp. 464–492.
7. W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. *ASIACRYPT 2019, Part I*, pp. 227–247.
8. W. Beullens and G. Seiler. Labrador: Compact proofs for r1cs from module-sis. Cryptology ePrint Archive, Paper 2022/1341.
9. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. *Advances in Cryptology – EUROCRYPT 2014*, pp. 533–556, Berlin, Heidelberg, 2014.
10. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. *EUROCRYPT 2016, Part II*, pp. 327–357.
11. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pp. 315–334.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO'94*, pp. 174–187.

13. L. De Feo, T. B. Fouotsa, P. Kutas, A. Leroux, S.-P. Merz, L. Panny, and B. Wesolowski. SCALLOP: Scaling the CSI-FiSh. In *PKC 2023, Part I*, pp. 345–375.
14. L. De Feo and M. Meyer. Threshold schemes from isogeny assumptions. *PKC 2020, Part II*, pp. 187–212.
15. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO'86*, pp. 186–194.
16. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Advances in Cryptology – CRYPTO 2013*, pp. 75–92, Berlin, Heidelberg, 2013.
17. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38:691–729.
18. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, page 469–477, New York, NY, USA, 2015.
19. J. Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Paper 2009/007.
20. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. *Advances in Cryptology - CRYPTO 2009*, pp. 192–208, Berlin, Heidelberg, 2009.
21. S. Katsumata, Y. Lai, J. T. LeGrow, and L. Qin. CSI -otter: Isogeny-based (partially) blind signatures from the class group action with a twist. *CRYPTO 2023*, pp. 729–761.
22. Y.-F. Lai. CAPYBARA and TSUBAKI: Verifiable random functions from group actions and isogenies. Cryptology ePrint Archive, Report 2023/182.
23. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Advances in Cryptology — CRYPTO '91*, pp. 129–140, Berlin, Heidelberg, 1992.
24. C. Peikert. He gives C-sieves on the CSIDH. *EUROCRYPT 2020, Part II*, pp. 463–492.
25. R. S. Wahby, I. Tzialla, a. shelat, J. Thaler, and M. Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pp. 926–943.
26. Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pp. 863–880.

Appendix

A Security of commitment schemes

We start by recalling the formal notion of hiding and binding security for commitments.

Definition 15 (Hiding security). *A commitment scheme is* hiding *if for any probabilistic polynomial time adversary $\mathcal{A}$ playing the* Hide_b *game the advantage is negligible i.e.*

$$\mathsf{Adv}^{\mathsf{Hide}}(\mathcal{A}) = |\Pr[\mathsf{Hide}_1(\mathcal{A}) \to 1] - \Pr[\mathsf{Hide}_0(\mathcal{A}) \to 1]| = \mathsf{negl}(\lambda)$$

where the Hide_b *game is defined as follows : Given public parameters* pp*,* $\mathcal{A}$ *outputs two challenge messages* m_0, m_1 *and a state* st*. The challenger samples some randomness* r*, and then computes* $\mathsf{com} = \mathsf{Commit}(m_b, r)$*. On input the state* st *and commitment* com*,* $\mathcal{A}$ *returns a bit* b' *corresponding to its guess for the value of* b*.*

Definition 16 (Binding security). *A commitment scheme is binding if for any probabilistic polynomial time adversary* $\mathcal{A}$ *playing the* Bind *game, the advantage is negligible,i.e.,*

$$\mathsf{Adv}^{\mathsf{Bind}}(\mathcal{A}) = \Pr[\mathsf{Bind}(\mathcal{A}) \to 1] = \mathsf{negl}(\lambda)$$

where the Bind *game is defined as follows : Given the public parameters* pp*,* $\mathcal{A}$ *must output two* different *messages* m_0, m_1 *and associated randomness* r_0, r_1 *and wins if* $\mathsf{Commit}(m_0, r_0) = \mathsf{Commit}(m_1, r_1)$*.*

A.1 Security of sigma-protocols

We also give formal definitions for the notions of honest-verifier zero-knowledge as well as 2-special soundness.

Definition 17 (Honest Verifier Zero-Knowledge (HVZK)). *We say* Π_Σ *is* honest-verifier-zero-knowledge *for a relation* $\mathcal{R}_{\mathsf{pp}}$ *if there exists a* PPT*simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle* $\mathcal{O}$ *such that for any statement-witness pair* $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}_{\mathsf{pp}}$*,* $\mathsf{chall} \in \mathsf{ChSet}$*,* $\lambda \in \mathbb{N}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most a polynomial number of queries to* $\mathcal{O}$*, the advantage* $\mathsf{Adv}^{\mathsf{HVZK}}_{\Pi_\Sigma}$ *of* $\mathcal{A}$ *is negligible, where*

$$\mathsf{Adv}^{\mathsf{HVZK}}_{\Pi_\Sigma}(\mathcal{A}) := \left|\Pr[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, \mathsf{chall})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, \mathsf{chall})) = 1]\right|,$$

where $P^{\mathcal{O}} = (P_1, P_2)$ *is a prover running on* $(\mathsf{st}, \mathsf{wt})$ *with a challenge fixed to* chall *and the probability is taken over the randomness used by* (P, V) *and by the random oracle.*

Definition 18 (2-Special Soundness). *We say a sigma protocol* Π_Σ *has* 2-special soundness *if there exists a polynomial-time extraction algorithm such that, given a statement* st *and any two valid transcripts* $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ *and* $(\mathsf{com}, \mathsf{chall}', \mathsf{resp}')$ *relative to* st *and such that* $\mathsf{chall} \neq \mathsf{chall}'$*, outputs a witness* wt *satisfying* $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}_{\mathsf{pp}}$*.*

A.2 Hard problems for group actions

We introduce three hard problems in the context of group actions.

Definition 19 (Group Action Inverse Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action with a distinguished element $E_0 \in \mathcal{E}$. Given E sampled uniformly in $\mathcal{E}$, the* Group Action Inverse Problem (GAIP) *consists in finding an element $g \in G$ such that $g \star E_0 = E$.*

Definition 20 (Computational Diffie-Hellman Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action with a distinguished element $E_0 \in \mathcal{E}$. The* computational Diffie-Hellman (CDH) problem *consists in, given a tuple $(g_1 \star E_0, g_2 \star E_0)$ where g_1, g_2 are sampled uniformly from G, computing $(g_1 + g_2) \star E_0$.*

Definition 21 (Decisional Diffie-Hellman Problem). *Let $(G, E_0, \mathcal{E}, \star)$ be a group action. The challenger generates $s_1, s_2, s_3 \in G$ and $b \in \{0,1\}$, and gives $(s_1 \star E_0, s_2 \star E_0, ((1-b)(s_1+s_2) + b(s_3)) \star E_0)$ to the adversary $\mathcal{A}$. Then, Decisional Diffie-Hellman (DDH) problem consists in $\mathcal{A}$ returning $b' \in \{0,1\}$ to guess b. The adversary $\mathcal{A}$ wins if $b = b'$. We define the advantage of $\mathcal{A}$ to be $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) := |\Pr[\mathcal{A}$ wins.$] - 1/2|$ where the probability is taken over the randomness of s_1, s_2, s_3, b.*

A.3 Proof Systems

We consider *non-interactive zero-knowledge proof of knowledge* (NIZKPoK) over the programmable random oracle model with statistical zero-knowledge and simulation-extractability.

Formally, a NIZKPoK for a relation $\mathcal{R}_{\mathsf{pp}}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Prove}, \mathsf{Verify})$ with access to the random oracle $\mathcal{O}$ such that for any valid public parameter of the scheme pp and the security parameter $\lambda \in \mathbb{N}$, the scheme satisfies the following properties:

- Completeness: for every $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}_{\mathsf{pp}}$,

$$\Pr[V^{\mathcal{O}}(\mathsf{st}, \pi) = 1 \mid \pi \leftarrow P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})] = 1.$$

- Statistical zero-knowledge: there exists a simulator Sim such that for any $(\mathsf{st}, \mathsf{wt}) \in \mathcal{R}_{\mathsf{pp}}$ and any possibly unbounded adversary $\mathcal{A}$ can distinguish the distributions: $\{\pi | \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})\}$, $\{\pi | \pi \leftarrow \mathsf{Sim}^{\mathcal{O}}(\mathsf{st})\}$ with only negligible advantage where the probability is taken over the randomness used in the experiment and the scheme.
- Simulation-extractability: for any adversary $\mathcal{A}$ there exists an extractor $\mathcal{E}_{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{r|r} \mathsf{Verify}(\mathsf{st}, \mathsf{wt}, \pi) = \top & (\mathsf{st}, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}, \mathcal{O}}(\mathsf{pp}) \\ (\mathsf{st}, \mathsf{wt}) \notin \mathcal{R}_{\mathsf{pp}} \ \wedge \ (\mathsf{st}, \mathsf{wt}, \pi) \notin L & \mathsf{wt} \leftarrow \mathcal{E}(\mathsf{st}, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{SimProve}(\mathsf{st}', \mathsf{wt}')$ is an oracle that returns a simulated proof using $\pi' \leftarrow \mathsf{Sim}(\mathsf{st}')$ when $(\mathsf{st}', \mathsf{wt}') \in R$. Otherwise, $\mathsf{SimProve}$ returns $\perp$. Also, $\mathsf{SimProve}$ maintains the list L, which is initially empty, and records $(\mathsf{st}', \mathsf{wt}', \pi')$ for each valid query of $(\mathsf{st}', \mathsf{wt}')$.

B Multi-Proof Online Extractability

The notion *multi-proof online extractability* [5] is defined as follows.

Definition 22 (Multi-Proof Online Extractability (mpOE)). *A* NIZK *proof system Π_{NIZK} is (multi-proof) online extractable if there exists a PPT extractor* $\mathsf{OnlineExtract}$ *such that for any (possibly computationally-unbounded) adversary $\mathcal{A}$ making at most polynomially-many queries has at most a negligible advantage in the following game played against a challenger (with access to a random oracle $\mathcal{O}$).*

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

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

- $(\mathtt{hash}, x)$*: The challenger updates $L_{\mathcal{O}} \leftarrow L_{\mathcal{O}} \cup \{(x, \mathcal{O}(x))\}$ and returns $\mathcal{O}(x)$. We assume below that $\mathcal{A}$ runs the verification algorithm after receiving a proof from the prover oracle and before submitting a proof to the extract oracle.*[6]
- $(\mathtt{prove}, \mathsf{lbl}, \mathsf{st}, \mathsf{wt})$*: The challenger returns $\bot$ if $\mathsf{lbl} \notin \mathsf{L}$ or $(\mathsf{st}, \mathsf{wt}) \notin \mathcal{R}_{\mathsf{pp}}$. Otherwise, it returns $\pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{st}, \mathsf{wt})$ and updates $L_{\mathsf{Prove}} \leftarrow L_{\mathsf{Prove}} \cup \{\mathsf{lbl}, \mathsf{st}, \pi\}$.*
- $(\mathtt{extract}, \mathsf{lbl}, \mathsf{st}, \pi)$*: The challenger checks if $\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{st}, \pi) = \top$ and $(\mathsf{lbl}, \mathsf{st}, \pi) \notin L_{\mathsf{Prove}}$, and returns $\bot$ if not. Otherwise, it runs $\mathsf{wt} \leftarrow \mathsf{OnlineExtract}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{st}, \pi, L_{\mathcal{O}})$ and checks if $(\mathsf{st}, \mathsf{wt}) \notin \mathcal{R}_{\mathsf{pp}}$, and returns $\bot$ if yes and sets $\mathsf{flag} = 1$. Otherwise, if all checks pass, it returns wt.*

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

Multi-proof online extractability provides a strong guarantee that an adversary cannot break the simulation extractability of the proof while the extraction is executed online.

C Proofs

C.1 Proof of Theorem 4

We give brief proofs for correctness, special soundness, statistical honest-verifier zero knowledge for the proof system in Fig. 2.

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

Proof. **Correctness.** For the challenge $\mathsf{chall} = 0$, the correctness holds naturally. For $\mathsf{chall} = 1$, for the response $\mathsf{resp} = (r'', r'')$, we have $r'' = r' + r$. Therefore, $(m'', r'') \star X = (m' + m, r' + r) \star X$. We have $\widetilde{\mathsf{C}} = \mathsf{C}$ and the correctness follows.

Special Soundness. Let $(\mathsf{comm} = \mathsf{C}, 0, \mathsf{resp}_0 = \mathsf{seed}), (\mathsf{comm} = \mathsf{C}, 1, \mathsf{resp}_1 = (m'', r''))$ be two valid transcripts. We show that can either extract a collision of the hash function H or a witness for the statement (pp, M', u).

Following **round 1** of Fig. 1, say seed generates m', r'. Due to the validity of both transcripts, we have $\mathsf{C} = H(\mathtt{COM}||(m'', r'') \star X) = H(\mathtt{COM}||(m', r') \star u)$. We assume $(m'', r'') \star X = (m', r') \star u$; otherwise, a collision of the hash function H can be found. It is clear that $(m'' - m, r'' - r)$ is the witness since $(m'' - m, r'' - r) \star X = u$. We therefore have the extractor.

Statistical HVZK. For the challenge $\mathsf{chall} = 0$, the simulator follows the first round of Fig. 2 and generates $(\mathsf{seed}, 0, \mathsf{comm})$ as the transcript. For the challenge $\mathsf{chall} = 1$, the simulator generates m'', r'' from $\mathcal{M}', \mathcal{R}'$ uniformly at random and computes $\widetilde{\mathsf{C}} = H(\mathtt{COM}||(m'', r'') \star X)$. The distributions $\{(m', r') \star u|(m', r') \leftarrow \mathcal{M}' \times \mathcal{R}'\}$ and $\{(m'', r'') \star X|(m'', r'') \leftarrow \mathcal{M}' \times \mathcal{R}'\}$ are identical for all u, hence the scheme is statistical HVZK. Concretely, since seed has λ bits of min-entropy and is information-theoretically hidden from the distinguisher, for any unbounded distinguisher with Q queries to H of the form $H(\mathtt{EXP}||\cdot)$ the statistical difference between the distributions is $Q_H/2^\lambda$.

C.2 Proof of Theorem 6

We give here the proof of soundness and honest-verifier zero-knowledge of Theorem 6.

Proof. **2-Special Soundness.** Let u be the statement and $(\mathsf{root}, 1, (M'', R'', \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1)), (\mathsf{root}, 0, (\mathsf{seed}, \mathsf{bits}_{b+1}, \mathsf{bits}_{b+2}, \mathsf{C}_2))$ be the two valid transcripts. We claim that we are able to extract a relation witness of the statement u for $\mathcal{R}_{\mathsf{pp}, A_0, A_0}$ or $\mathcal{R}_{\mathsf{pp}, A_1, A_1}$, a collision witness for $\mathsf{MerkleTree}(\cdot)$ or a binding witness for $H(\mathtt{COM}||\cdot)$.

Due to the validness of the proof, we have

$$\mathsf{root} = \mathsf{MerkleTree}((\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}'), (\widetilde{\mathsf{C}}'||\widetilde{\mathsf{C}}||\mathsf{C}_1)) = \mathsf{MerkleTree}((\widehat{\mathsf{C}}||\widehat{\mathsf{C}}'||\mathsf{C}_2), (\mathsf{C}_2||\widehat{\mathsf{C}}||\widehat{\mathsf{C}}')),$$

where

$$\begin{aligned} \widetilde{\mathsf{C}} &\leftarrow H(\mathtt{COM}||(M'', A_0 M'', R'') \star (X, X)||\mathsf{bits}_0), \\ \widetilde{\mathsf{C}}' &\leftarrow H(\mathtt{COM}||(M'', A_1 M'', R'') \star (X, X)||\mathsf{bits}_2), \\ (M', R') &\leftarrow H(\mathtt{EXP}||\mathsf{seed}), \\ \widehat{\mathsf{C}} &\leftarrow H(\mathtt{COM}||(M', A_0 M', R') \star u||\widehat{\mathsf{bits}}), \\ \widehat{\mathsf{C}}' &\leftarrow H(\mathtt{COM}||(M', A_1 M', R') \star u||\widehat{\mathsf{bits}}'). \end{aligned}$$

If the alphabetically ordered set $\{(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}'), (\widetilde{\mathsf{C}}'||\widetilde{\mathsf{C}}||\mathsf{C}_1)\}$ is not equal to $\{(\widehat{\mathsf{C}}||\widehat{\mathsf{C}}'||\mathsf{C}_2), (\mathsf{C}_2||\widehat{\mathsf{C}}||\widehat{\mathsf{C}}')\}$, then these two ordered set constitute a witness of collision of $\mathsf{MerkleTree}(\cdot)$. Therefore, we may assume

$$\{(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}'), (\widetilde{\mathsf{C}}'||\widetilde{\mathsf{C}}||\mathsf{C}_1)\} = \{(\widehat{\mathsf{C}}||\widehat{\mathsf{C}}'||\mathsf{C}_2), (\mathsf{C}_2||\widehat{\mathsf{C}}||\widehat{\mathsf{C}}')\}.$$

We consider two cases: $(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}') = (\widehat{\mathsf{C}}||\widehat{\mathsf{C}}'||\mathsf{C}_2)$ or $(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}') = (\mathsf{C}_2||\widehat{\mathsf{C}}||\widehat{\mathsf{C}}')$.

For each case, we will extract a witness for either the relations or the binding commitment $H(\mathtt{COM}||\cdot)$.

- **Case** $(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}') = (\widehat{\mathsf{C}}||\widehat{\mathsf{C}}'||\mathsf{C}_2)$**:** Due to $\widetilde{\mathsf{C}} = \widehat{\mathsf{C}}$, if the equality $((M'', A_0M'', R'')\star(X, X)||\mathsf{bits}_0) = ((M', A_0M', R')\star u||\widehat{\mathsf{bits}})$ does not hold, then the pairs serve as a binding witness for $H(\mathtt{COM}||\cdot)$. If the equality holds then we have $(u, ((M, A_0M), R)) \in \mathcal{R}_{\mathsf{pp},A_0,A_0}$ where $M = M'' - M'$ and $R = R'' - R'$.
- **Case** $(\widetilde{\mathsf{C}}||\mathsf{C}_1||\widetilde{\mathsf{C}}') = (\mathsf{C}_2||\widehat{\mathsf{C}}||\widehat{\mathsf{C}}')$**:** Due to $\widetilde{\mathsf{C}}' = \widehat{\mathsf{C}}'$, if the equality $((M'', A_1M'', R'')\star(X, X)||\mathsf{bits}_2) = ((M', A_1M', R')\star u||\widehat{\mathsf{bits}'})$ does not hold, then the pairs serve as a binding witness for $H(\mathtt{COM}||\cdot)$. If the equality holds then we have $(u, ((M, A_1M), R)) \in \mathcal{R}_{\mathsf{pp},A_1,A_1}$ where $M = M'' - M'$ and $R = R'' - R'$.

Therefore, the special soundness follows.

HVZK. Given a parameter pp and a statement u, a simulator $\mathcal{S}$ on input $(\mathsf{pp}, u, \mathsf{chall})$ proceeds as Fig. 5.

We claim that for the transcripts generated by the simulator $\mathcal{S}$ are statistically indistinguishable from those generated by a real prover.

Let $((M_1, M_2), R)$ be the witness for u. When $\mathsf{chall} = 1$, two transcript distributions are identical except for two differences. First, there exist no $\mathsf{seed} \in \{0,1\}^\lambda$ has been queried to $H(\mathtt{EXP}||\cdot)$ such that $(M'', R'') = (M_1 + M', R + R')$ where M', R' are generated by the oracle. Also, there exists no $\mathsf{bits}_1 \in \{0,1\}^\lambda$ has been queried to $H(\mathtt{COM}||\cdot)$ such that $\mathsf{C}_1 = H(\mathtt{COM}||u'||\mathsf{bits}_1)$ for some set element $u' \in \mathcal{C}'^2$. Since we model H as a random oracle, the statistical difference in this case is at most

$$\left|\frac{2Q_H}{2^\lambda} + \frac{Q_H}{|\mathcal{R}'^2|}\right|.$$

Similarly, when $\mathsf{chall} = 0$, two transcript distributions are identical except that there exist no $\mathsf{bits}_2 \in \{0,1\}^\lambda$ has been queried to $H(\mathtt{COM}||\cdot)$ such that $\mathsf{C}_2 = H(\mathtt{COM}||u'||\mathsf{bits}_2)$ for some set element $u' \in \mathcal{C}'^2$. Since we model H as a random oracle, the statistical difference in this case is at most

$$\left|\frac{Q_H}{2^\lambda}\right|.$$

It follows that the statistical difference of the transcripts generated by $\mathcal{S}$ and a real prover is at most

$$\left|\frac{2Q_H}{2^{2\lambda}} + \frac{Q_H}{|\mathcal{R}'^2|}\right|.$$

```
$\mathcal{S}^{\mathcal{O}}(\mathsf{pp}, \mathsf{st} = u, \mathsf{chall})$
if $\mathsf{chall} = 1$ then
    $(M'', R'', \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1) \leftarrow \mathcal{M}' \times \mathcal{R}'^2 \times \{0,1\}^\lambda \times \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$
    $\mathsf{C}_0 \leftarrow H(\mathtt{COM}||(M'', A_0 M'', R'') \star (X, X)||\mathsf{bits}_0)$
    $\mathsf{C}_2 \leftarrow H(\mathtt{COM}||(M'', A_1 M'', R'') \star (X, X)||\mathsf{bits}_2)$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{MerkleTree}((\mathsf{C}_0||\mathsf{C}_1||\mathsf{C}_2), (\mathsf{C}_2||\mathsf{C}_0||\mathsf{C}_1))$    ▷ Index-hiding Merkle Tree
else
    $(\mathsf{seed}, \mathsf{bits}_0, \mathsf{bits}_2, \mathsf{C}_1) \leftarrow \{0,1\}^\lambda \times \{0,1\}^\lambda \times \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$
    $(M', R') \leftarrow H(\mathtt{EXP}||\mathsf{seed})$
    $\mathsf{C}_0 \leftarrow H(\mathtt{COM}||(M', A_0 M', R') \star u||\mathsf{bits}_0)$
    $\mathsf{C}_1 \leftarrow H(\mathtt{COM}||(M', A_1 M', R') \star u||\mathsf{bits}_1)$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{MerkleTree}((\mathsf{C}_0||\mathsf{C}_1||\mathsf{C}_2), (\mathsf{C}_2||\mathsf{C}_0||\mathsf{C}_1))$
$\mathsf{resp} \leftarrow (\mathsf{root}, \mathsf{chall}, \mathsf{resp})$
return $\pi \leftarrow (\mathsf{com}, \mathsf{chall}, \mathsf{resp})$
```

Figure 5. The simulator the protocol described in Fig. 4.