Schnorr Blind Signatures and Signed ElGamal KEM in Algebraic Group Action Model

Dung Hoang Duong[1], Willy Susilo[1], and Chuanqi Zhang[2]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia.
{hduong,wsusilo}@uow.edu.au
[2] Department of Software Systems & Cybersecurity, Faculty of Information Technology, Monash University, Australia.
chuanqi.zhang@monash.edu

Abstract. Schnorr blind signature is one of the most efficient and widely used blind signatures. At CRYPTO'23, Katsumata et al. proposed CSI-Otter, the first blind signature from isogenies, which does not follow the construction framework of the Schnorr blind signature. Instead, CSI-Otter was constructed from the sigma protocol for an OR relation that captures the idea of the Abe-Okamoto signature and hence can adapt the proof techniques by Kastner, Loss and Xu (PKC'22) into its security proof. Unfortunately, the concurrent security of CSI-Otter was later broken independently by Katsumata et al. (PKC'24) and Do et al. (Eurocrypt'24). As a result, CSI-Otter and Schnorr-like blind signature schemes constructed from Sigma protocols with small challenge space should not be used for polynomially many concurrent signing sessions without additional boosting transformations. Sequential security, and in some settings logarithmic-session concurrent security, remain meaningful security guarantees.

In this paper, we provide an intensive study of the Schnorr blind signature from isogenies in the Algebraic Group Action Model (AGAM) and the Random Oracle Model (ROM). In particular, we first prove the tight security of the existing Schnorr signature from isogenies under the group action discrete logarithm assumption (GA-DLOG) in AGAM + ROM, which serves as the foundation for the proof of the sequential security and logarithmic-session concurrent security of the Schnorr blind signature in AGAM + ROM under the hardness of the one-more group action discrete logarithm (OM-GA-DLOG) assumption. We also clarify a limitation of the direct parallel-repetition approach: because the large challenge space is obtained from binary challenges, the construction should not be claimed to satisfy general two-open-session concurrent security for polynomially many total signing sessions. In addition, of independent interest, we also present the Schnorr-Signed Hashed ElGamal KEM from isogenies and prove its CCA2 security in AGAM + ROM under the hardness of GA-DLOG.

1 Introduction

Schnorr Blind Signature. Blind signature was introduced by Chaum [19], which allows a user to obtain a valid signature on a message from a signer while keeping the message blinded to the signer. A secure blind signature needs to satisfy two properties: (1) *blindness*: the signer cannot link the transcripts of protocol runs to the signatures that they created, particularly, it does not learn the message it signed; and (2) *one-more unforgeability*: the user cannot create $\ell + 1$ or more valid signatures after only ℓ interactions with the signer. Those special features have made blind signatures available in many applications, including e-cash systems ([19,21,36,60]), anonymous credentials ([15,17,6], and electronic voting ([20,35]).

Schnorr blind signature [22] is one of the most efficient blind signatures, which is built from the Schnorr signature [50]. The security of the Schnorr signature has been analyzed in the random oracle model (ROM) under the discrete logarithm (DL) assumption [48]. However, their proofs rely on the Forking Lemma, which involves rewinding the adversary and leads to a loose reduction: the success probability of the discrete logarithm solver is reduced by a factor of q_h, the number of random oracle queries made by the adversary. In contrast, the security of the Schnorr blind signature remains less well understood. Several works [5,32,47] suggested that Schnorr blind signatures cannot be proven one-more unforgeable under standard assumptions – even in the random oracle model (ROM) until Fuchsbauer-Plouviez-Seurin [34] who provided a proof for the Schnorr blind signature in the algebraic group model (AGM) [33]. In particular, they first proved the tight security of the Schnorr signature from the DL assumption in AGM + ROM and also proved that the Schnorr blind signature is secure in AGM + ROM assuming hardness of the one-more discrete logarithm (OMDL) problem and ROS [51] (when the signing sessions are running concurrently). However, Benhamouda et al. [8] discovered a polynomial-time algorithm for solving ROS, which showed that the Schnorr blind signature cannot achieve concurrent security. Later, at PKC 2022, Kastner-Loss-Xu [40] provided a proof that the Schnorr blind signature is secure in AGM + ROM under the OMDL assumption if the signatures are issued *sequentially*. It is also argued by Kastner et al. [40] that *security under sequential signing sessions is still a very meaningful security guarantee and has been explored in prior works*; please refer to [40] for a detailed discussion.

Recently, Harding and Xu [39] provided a study of the Schnorr blind signature in a limited concurrency setting. In particular, they proved that the Schnorr blind signature is 2-concurrently secure, i.e., when at most 2 signing sessions can be opened concurrently (henceforth the 2-concurrency setting), in AGM + ROM, assuming the hardness of OMDL. Their work motivates the study of limited concurrency. However, in our group-action setting the large challenge space is obtained by parallel repetition of a binary-challenge protocol. This distinction is important: mix-and-match attacks by Do et al. [26] against repeated Schnorr-type blind signatures show that allowing only two sessions to be open at a time is still not enough when the total number of signing sessions reaches the repetition

length. Therefore, in this work we restrict our positive one-more unforgeability claims to the sequential setting and to a logarithmic-session concurrent setting.

Cryptographic Group Actions. Cryptographic group actions were first introduced by Brassard and Yung [16] in the context of *one-way* group actions. Couveignes [25] later formalized *hard homogeneous spaces* and proposed examples stemming from complex multiplication of elliptic curves, while Rostovtsev and Stolbunov [49] further developed isogeny-based cryptographic constructions in this direction. Cryptographic group actions have recently attracted significant attention as promising tools for post-quantum cryptography. In particular, several candidates submitted to NIST's call for additional digital signatures, whose first-round submission deadline was in June 2023 [46], are based on cryptographic group actions, including LESS [12,4], MEDS [24,23], ALTEQ [58,13], and HAWK [27,14]. Notably, all of these are based on non-commutative group actions.

The leading example of a cryptographic group action to date is the CSIDH protocol [18], which realizes a commutative group action via isogenies between supersingular elliptic curves defined over a finite field. Due to the commutativity of the underlying groups, since then, there have been many works for cryptographic designs, including digital signature CSI-FiSh [11], ring and group signatures [10], password authenticated key exchange [1], threshold signatures [30], and blind signatures [42].

Blind Signatures from Group Actions. The first blind signature from CSIDH-based group actions was introduced by Katsumata et al. [42], called CSI-Otter. Their construction follows the Schnorr paradigm, but it is built from a Sigma protocol for an OR relation in which the signer knows one of two secrets corresponding to the public key. This allows them to adapt the proof techniques of Kastner et al. [40] for the Abe–Okamoto partially blind signature [2] to prove the security of CSI-Otter. Subsequently, Kutcha et al. [44] proposed a blind signature from MEDS [23], and Duong et al. [29] generalized the design of CSI-Otter to construct blind signatures from non-commutative group actions. They also gave an instantiation based on a variant of linear code equivalence related to LESS [4], interpreted as a symmetric group action on equivalence classes. In contrast, our goal in this paper is a more direct Schnorr-style construction whose final signature has the same compact Schnorr form as the underlying signature.

Unfortunately, due to the restricted structure of group actions, the challenge spaces of the Sigma protocols underlying signatures from group actions are small, e.g., $\{0, 1\}$ or $\{-1, 1\}$. Therefore, the protocol must be repeated many times to achieve the desired security level, which leads to large signatures. This also creates a weakness for the existing blind signatures mentioned above. In particular, Katsumata et al. [43] and Do et al. [26] proposed attacks against the concurrent security of blind signatures based on identification protocols that use parallel repetition to reduce soundness error, including CSI-Otter [42], [44], and [29]. As a result, these existing blind signatures from group actions do not support polynomially many concurrent sessions in their basic form. They are therefore

typically considered in the sequential setting, unless an additional transformation is applied.

Very recently, Hanzlik et al. [37] proposed several general frameworks, Tanuki, for blind signatures from cryptographic group actions, which in particular admit instantiations from isogenies and other post-quantum assumptions. Their frameworks are broader in scope than ours, and include a construction achieving security under polynomially many concurrent sessions while relying on a non-standard one-more vectorization assumption. Subsequently, Hanzlik, Lai, Paracucchi, and Persichetti introduced WOMBAT [38], a new framework building on Tanuki's techniques that achieves polynomially concurrent security in the algebraic group action model under the standard group action inverse problem, with instantiations based on CSIDH and LESS. The goal of these works is to develop general group-action frameworks with strong concurrent security guarantees, rather than to include the direct Schnorr/CSI-FiSh-style blind signature analogue as an object of study.

By contrast, our focus in this paper is different and complementary. We study a minimal Schnorr-style construction in the static-key setting for twist-enabled action families. In the CSIDH setting, this yields a particularly simple and compact blind signature design that closely mirrors the underlying Schnorr/CSI-FiSh-style signature. We prove sequential and logarithmic-session concurrent security, but we do not claim polynomial-session concurrent security for this direct Schnorr-style construction.

The benefit of a Schnorr blind signature in the classical setting is that the signature has exactly the same form and size as the underlying Schnorr signature, which is efficient, simple, and compatible with standardized Schnorr-type signatures such as EdDSA. While the digital signature from the CSIDH action, e.g., CSI-FiSh [11], is of Schnorr type, there is still no direct Schnorr-style blind signature from CSIDH-type group actions with the same compact form as the underlying Schnorr/CSI-FiSh signature. The signature size of CSI-Otter is twice that of CSI-FiSh. Hence, it is natural to raise the following question.

Can we construct an efficient Schnorr blind signature from isogenies? If yes, what one-more unforgeability guarantee can we prove?

1.1 Our Contributions

In this paper, we provide an affirmative answer to the above question. In particular, we construct a Schnorr blind signature scheme from CSIDH group actions and prove its security under the *one-more group action discrete logarithm* (OMGA-DLOG) assumption in the *algebraic group action model* (AGAM) [28] and the random oracle model (ROM). Our work is inspired by the previous work by Fuchsbauer et al. [34], Kastner et al. [40], and Harding and Xu [39] for Schnorr blind signature in the classical settings. Our contributions in this work can be summarized as follows.

– First, following Fuchsbauer et al. [34], we show that the Schnorr signature from group actions (e.g., CSI-FiSh [11]) is tightly secure under the *group*

action discrete logarithm (GA-DLOG) assumption in AGAM + ROM. Its security proof is a foundation for the proof of one-more unforgeability of the Schnorr blind signature.
– We then provide the design of the Schnorr blind signature from isogenies[3] and provide a detailed study on its one-more unforgeability. In particular, we prove its sequential security and logarithmic-session concurrent security in AGAM + ROM under the hardness of *one-more group action discrete logarithm* (OM-GA-DLOG) problem that generalizes the OMDL in the classical settings to group actions. As a result, the signature size of our Schnorr blind signature is the same as the Schnorr signature (e.g., CSI-FiSh), which is half that of CSI-Otter [42]. We do not claim general two-open-session concurrent security for polynomially many total signing sessions. This is because the construction uses parallel repetition of a binary-challenge protocol, and this structure is vulnerable to mix-and-match attacks once the total number of sessions reaches the repetition length.
– Of independent interest, we provide a construction of Schnorr-Signed Hashed ElGamal KEM (as akin to [34] for the classical settings) from group actions and prove its tight CCA2-security in AGAM + ROM under the hardness of GA-DLOG. A previously known result is the ElGamal KEM from isogenies by Duman et al. [28] with CCA1-security in AGAM + ROM under the hardness of *Group Action q-Decisional Diffie-Hellman* problem.

1.2 Technical Overview

We now explain our contributions in detail. We will first review how the Schnorr signature from isogenies, such as CSI-FiSh [11], is constructed and then how it is extended to a Schnorr blind signature. We then review the tools and techniques for the security proofs.

Schnorr blind signature from isogenies. The CSIDH group action [18] is defined as $\diamond : \mathcal{G} \times \mathcal{E} \to \mathcal{E}$ where $\mathcal{G}$ is the ideal class group and $\mathcal{E}$ is the set of elliptic curves, and we further assume that the structure of $\mathcal{G}$ is known and can be expressed as $\mathcal{G} \cong \mathbb{Z}_N$ for some $N \in \mathbb{N}$; here for $a \in \mathbb{Z}_N$ and $E \in \mathcal{E}$, we write $a \diamond E$ for the action of a on E. Note that for $a, b \in \mathbb{Z}_N$ we have $a \diamond (b \diamond E) = (a+b) \diamond E$.

The underlying identification scheme of CSI-FiSh between the prover with $(\mathsf{pk}, \mathsf{sk}) = (E_1 := a \diamond E_0, a)$ for a public $E_0 \in \mathcal{E}$ and $a \xleftarrow{\$} \mathbb{Z}_N$ with the verifier with $\mathsf{pk} = E_1$ is as follows. The prover first samples $y \in \mathbb{Z}_N$ and computes the commitment $Y := y \diamond E_0$ and sends it to the verifier who replies with a challenge $c \in \{0, 1\}$. The prover then computes the response $r := y - c \cdot x$ and sends it to the verifier who can verify by checking whether $r \diamond E_c \stackrel{?}{=} Y$. The protocol

[3] Note that this design was sketched by Katsumata et al. in [42] but the authors mentioned that Schnorr blind signature is not known to be secure in ROM and hence the Schnorr blind signature in the isogeny setting. This made Katsumata et al. investigate the Abe-Okamoto blind signatures [2] instead and adapt the proof by Kastner et al. [41] to the isogeny setting.

has soundness error $\frac{1}{2}$ and hence needs to be repeated many times to attain the required security level. The Schnorr signature is hence obtained by applying the Fiat-Shamir's transform [31] by using a hash function H, modeled as a random oracle, to compute $c := \mathsf{H}(Y\|m)$. The signature for the message m is $\sigma := (Y, r)$ which can be verified by first computing $c = \mathsf{H}(Y\|m)$ and then checking whether $r \diamond E_c \stackrel{?}{=} Y$. The signature can be reduced to (c, r) and verification can be done by first computing $Y := r \diamond E_c$ and checking whether $c \stackrel{?}{=} \mathsf{H}(Y\|m)$.

In order to extend to Schnorr blind signature from the CSIDH group action, one needs to rely on the property that isogenies are more expressive than a normal group action due to *quadratic twist*. Given $E_1 = a \diamond E_0$ for unknown $a \in \mathbb{Z}_N$, we can efficiently compute its quadratic twist $(-a)\diamond E_0$ without knowing a. For convenience, we denote the twist of E_1 by E_{-1}. We also use E^{-1} for the twist of E. Once using twists, then the previous challenge space $\{0, 1\}$ should change to $\{-1, 1\}$. The interaction between the signer and user in the Schnorr blind signature works as follows.

In the first round, the signer samples $y \xleftarrow{\$} \mathbb{Z}_N$ and computes the commitment $Y := y \diamond E_0$ and sends it to the user. The user now needs to respond with a challenge that *blinds* the message m. The user first samples $(d, w) \xleftarrow{\$} \{-1, 1\} \times \mathbb{Z}_N$ and computes $Y' := w \diamond Y^d$. It will then mask the message m by computing $c' := \mathsf{H}(Y'\|m)$, compute the challenge $c := d \cdot c'$ and send it to the signer, who will respond with $r := y - c \cdot x$. Upon receiving r, the user will check whether $r \diamond E_1^c \stackrel{?}{=} Y$. If yes, then it outputs the signature for m as (Y', r') where $r' = d \cdot r + w$. Alternatively, as above, the signature can be (c', r'), which is just half of that of CSI-Otter. As mentioned above, for a secure scheme, we need to enlarge the challenge space to $\{-1, 1\}^\lambda$ for a security parameter λ, and we will denote the elements are in bold as vectors, i.e., $\mathbf{c} \in \{-1, 1\}^\lambda$ is a vector $\mathbf{c} = (c_1, \ldots, c_\lambda)$ with $c_i \in \{-1, 1\}$ for $i \in [1, \ldots, \lambda]$. We will also extend this to $\mathbf{Y} \in \mathcal{E}^\lambda$ and $\mathbf{y} \in \mathbb{Z}_N^\lambda$, respectively.

Security proof techniques. The challenge arises when it comes to producing a security proof of one-more unforgeability for the Schnorr blind signature. As in the classical setting, no such proof is known in the generic group model (GGM). This is also why Katsumata et al. [42] adopted the design of Abe-Okamoto-like blind signature [2] in the isogeny setting, where they can utilize the proof techniques from Kastner et al. [41] for the Abe-Okamoto scheme. Security proof for Schnorr blind signature only exists in the algebraic group model (AGM), which was introduced by Fuchsbauer et al. [34] and then Kastner et al. [40]. In this work, we will follow this route.

First of all, we need to investigate the algebraic group model for group actions. The algebraic group model (AGM) was introduced by Fuchsbauer, Kiltz and Loss [33], which is a model that lies between the standard model and the generic group model (GGM) [54,45]. In the AGM, any adversary must output an explanation/representation of how it computed its output group elements from the group elements in its input. Since its introduction, AGM has been a useful tool to provide many security proofs, especially when they are hard to be ana-

lyzed in the plain model. In [28], Duman et al. generalized the notion to group actions, named *Algebraic Group Action Model (AGAM)*, and the AGAM with twists - which are relevant to many practical group action instantiations, like in our work. We hereby denote by AGAM the algebraic group action model with twists in this paper without confusion. Unlike the AGM, the adversary in the AGAM needs to output an explanation of how it computed each of its output elliptic curves from exactly one of its inputs. Concretely, if $E_1, \ldots, E_k$ are its received elliptic curves, when the adversary outputs $E \in \mathcal{E}$, it has to provide an element $a \in \mathbb{Z}_N$, an index $i \in \{1, \ldots, k\}$ and a $b \in \{-1, 1\}$ such that $E = a \diamond E_i^b$.

Let's sketch how to provide the tight security proof for the Schnorr signature in AGAM first, which serves as the foundation for the proof of one-more unforgeability of the Schnorr blind signature. For simplicity, let's consider the base case with a small challenge space $\{-1, 1\}$. Assume that the adversary has made q signature queries to obtain $(Y_1, s_1), \ldots, (Y_q, s_q)$ and output a forgery (Y^*, s^*). Since the adversary is algebraic, it also provides $(z, i, b) \in \mathbb{Z}_N \times \{1, \ldots, q\} \times \{-1, 1\}$ such that $Y^* = z \diamond Y_i^b$. Since (Y_i, s_i) is the valid signature returned to a query of the adversary by the challenger, one has $Y_i = s_i \diamond E_1^{c_i}$ with $c_i := \mathsf{H}(Y_i \| m_i) \in \{-1, 1\}$. It follows that $Y^* = z \diamond Y_i^b = (z + bs_i) \diamond E_1^{bc_i} = (z + bs_i + bc_i \cdot x) \diamond E_0$. In addition, (Y^*, s^*) is a valid forgery for m^*, one has $Y^* = s^* \diamond E_1^{c^*} = (s^* + c^* \cdot x) \diamond E_0$ where $c^* := \mathsf{H}(Y^* \| m^*) \in \{-1, 1\}$. From those equations, we obtain

$$Y^* = (z + bs_i + bc_i x) \diamond E_0 = (s^* + c^* \cdot x) \diamond E_0.$$

If the event $c^* \neq bc_i$ happens, then one can obtain the secret $x = \frac{z + bs_i - s^*}{c^* - bc_i}$, which is a solution for GA-DLOG. We will argue that in the real scheme where the challenge space is $\{-1, 1\}^\lambda$, the probability of this event is non-negligible; see Section 4 for the details.

It gets more complicated when we look at the proof of security for the Schnorr blind signature. Since we would like to adopt the techniques from Kastner et al. [40], we first need a version of one-more discrete logarithm (OMDL) problem for group actions, which we call *one-more group action discrete logarithm (OM-GA-DLOG)*, as defined in [37]. Roughly speaking, an adversary against ℓ-OM-GA-DLOG needs to find the group action discrete logarithm of $(\ell + 1)$ elements by making at most ℓ calls to an oracle, called gadlog, solving the group action discrete logarithm problem. We note that our gadlog oracle is different from the one in classical settings. In the classical settings with a public group element g, the discrete logarithm solver oracle, for a query input h, will return $y \in \mathbb{Z}_p$ such that $h = g^y$. In our CSIDH group action setting, given two inputs E_1, E_2, the oracle gadlog will return $a \in \mathbb{Z}_N$ such that $E_1 = a \diamond E_2$.

Let's sketch the proof of sequential security of the Schnorr blind signature. since the adversary can query the signing oracle $\mathsf{BS.Sign}_2$[4] a total of ℓ times, it is allowed to open $\ell+1$ sessions and close the first ℓ of them. Let $Y_1, \ldots, Y_{\ell+1}$ be the

[4] In blind signatures, the signer will consist of two algorithms $(\mathsf{BS.Sign}_1, \mathsf{BS.Sign}_2)$ in which $\mathsf{BS.Sign}_1$ will be used in the first round to create the commitment and $\mathsf{BS.Sign}_2$ will use the secret key to generate the signature for the masked challenge. Similarly, the user also consists of two algorithms $(\mathsf{BS.User}_1, \mathsf{BS.User}_2)$.

commitments returned by $\mathsf{BS.Sign}_1$, and let $(m_1, R_1, r_1), \ldots, (m_{\ell+1}, R_{\ell+1}, r_{\ell+1})$ be the output of the adversary, i.e., (R_i, r_i) is the forgery of m_i for $i \in \{1, \ldots, \ell+1\}$. Since the adversary is algebraic, for each $i \in \{1, \ldots, \ell+1\}$, it needs to output (j, z_{i_j}, b_{i_j}) such that either $R_i = z_{i_j} \diamond E_1^{b_{i_j}}$ or $R_i = z_{i_j} \diamond Y_j^{b_{i_j}}$. We consider only the latter case here. Since (R_i, r_i) is the forgery of m_i, we have $R_i = r_i \diamond E_1^{c_i^*}$ with $c_i^* = \mathsf{H}(R_i \| m_i)$. Two equations imply that $r_i \diamond E_1^{c_i^*} = z_{i_j} \diamond Y_j^{b_{i_j}} (*)$. The reduction $\mathcal{B}$ to $\mathsf{OM\text{-}GA\text{-}DLOG}$ works as follows. $\mathcal{B}$ queries its challenge oracle $\ell + 1$ times to obtain $E_1, Y_1, \ldots, Y_\ell$, samples $y_{\ell+1} \xleftarrow{\$} \mathbb{Z}_N$ and sets $Y_{\ell+1} := y_{\ell+1} \diamond E_0$. Then $\mathcal{B}$ simulates $\mathsf{BS.Sign}_1$ by returning $Y_1, \ldots, Y_{\ell+1}$. In order to simulate $\mathsf{BS.Sign}_2(c_i)$ query, $\mathcal{B}$ queries its gadlog oracle ℓ times and returns r_i such that $r_i \diamond E_1^{c_i} = Y_i$. Substituting into equation $(*)$ we get

$$(r_i + c_i^* \cdot x) \diamond E_0 = z_{i_j} \diamond (r_j \diamond E_1^{c_j})^{b_{i_j}} = (z_{i_j} + b_{i_j} \cdot r_j + b_{i_j} \cdot c_j \cdot x) \diamond E_0.$$

We then obtain the secret key $x := \frac{z_{i_j} + b_{i_j} \cdot r_j - r_i}{c_i^* - b_{i_j} \cdot c_j}$ as soon as the event $c_i^* \neq b_{i_j} c_j$ happens. In the real scheme with large challenge space $\{-1, 1\}^\lambda$, we will show that the probability that this event happens is non-negligible. Once obtaining x, $\mathcal{B}$ can return x and $r_i^* := r_i + c_i \cdot x$ as a discrete logarithm for Y_i for each $i = 1, \ldots, \ell$ as a solution for the $\mathsf{OM\text{-}GA\text{-}DLOG}$ problem.

The proof for logarithmic-session concurrent security follows the same high-level extraction strategy as in the sequential case, but the main difficulty is to bound the probability that the adversary can correlate the final random-oracle challenges with challenges used in signing sessions, i.e., the probability of matching events of the form $c_i^* = b_{i_j} c_j$. We reduce this bad event to the so-called *binary parallel Random Inhomogeneities in an Overdetermined* (pROS) problem, of which we will also provide the hardness complexity.

This logarithmic-session result should not be interpreted as a two-open-session security theorem for polynomially many total sessions. The reason is that the construction obtains the large challenge space $\{-1, 1\}^\lambda$ by parallel repetition of binary-challenge components. A mix-and-match adversary [26] can exploit this repeated structure by combining real signer responses and simulated transcript coordinates. Thus, the positive security claims below are restricted to the sequential and logarithmic-session settings.

It is worth noting that in the real scheme with a large challenge space $\{-1, 1\}^\lambda$, the reduction is more involved, since the outputs (z_{i_j}, j, b_{i_j}) now become a tuple of vectors, say $(\mathbf{z}, I, \mathbf{b})$. Especially, the index I will be the set of λ tuples (j, k_j) where $j \in \{1, \ldots, \ell+1\}$ and $k_j \in \{1, \ldots, \lambda\}$, i.e., the tuple $(I, \mathbf{z}, \mathbf{b})$ does not correspond to one commitment tuple $\mathbf{Y}$ but will correspond to a tuple of commitments that may appear in any positions in the set of commitment tuples $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_{\ell+1}\}$. In addition, we have to make $\ell\lambda$ queries to the gadlog oracle while given $\ell\lambda + 1$ challenges, which is still a sample of the $\mathsf{OM\text{-}GA\text{-}DLOG}$ problem; please refer to Section 5 for more details.

2 Preliminaries

2.1 Notations

We denote by $\mathbb{N}$ and $\mathbb{Z}$ the set of natural numbers and integers, respectively. We define the ring of integers modulo N is $\mathbb{Z}_N$ with representatives in $\left[-\frac{q}{2}, \frac{q}{2}\right) \cap \mathbb{Z}$. For a positive integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. We use bold letters for vectors, e.g., $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$. For a finite set S, we write $s \xleftarrow{\$} S$ for uniformly sampling s at random over S. We use $\cdot$ to denote the component-wise multiplication of vectors in $\mathbb{R}$, e.g., for $c \in \mathbb{R}, \mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, we write $c \cdot \mathbf{a}$ for $(c \cdot a_1, \ldots, c \cdot a_n)$ and $\mathbf{a} \cdot \mathbf{b}$ for $(a_1 \cdot b_1, \ldots, a_n \cdot b_n)$. With a slight abuse of such notation, we also write $E^{\mathbf{c}}$ for the vector $(E^{c_1}, \ldots, E^{c_n})$, where E stands for an elliptic curve; please see Section 3.2 for more details.

2.2 Signatures

Definition 1 (Signature). *A signature scheme SIG consists of the following algorithms.*

- $\mathsf{par} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$*: the setup algorithm takes as input the security parameter λ in unary and outputs public parameters par.*
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SIG.Setup}(\mathsf{par})$*: the key generation algorithm takes public parameter par as input and generates a secret key sk and a public key pk.*
- $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$*: the signature generation takes as input the secret key sk and a message m, outputs a signature σ.*
- $0/1 \leftarrow \mathsf{SIG.Ver}(\mathsf{pk}, m, \sigma)$*: the verification algorithm takes as input the public key pk, a message m with its signature σ, outputs 1 if the signature is valid and 0 otherwise.*

A signature is required to be complete and to satisfy existential unforgeability under chosen-message attack ($\mathsf{EUF\text{-}CMA}$), as defined below.

Definition 2 (Correctness). *A signature scheme SIG is correct if for any λ and message m, we have that*

$$\Pr\left[\mathsf{SIG.Ver}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{SIG.Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SIG.Setup}(\mathsf{par}) \\ \sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m) \end{array}\right] = 1.$$

Definition 3 ($\mathsf{EUF\text{-}CMA}$). *For a signature scheme SIG, define the unforgeability game $\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda)$ between a challenger and an adversary $\mathcal{A}$ is defined as in Fig 1.*

We denote the advantage of an adversary $\mathcal{A}$ in forging signatures for SIG in the $\mathsf{EUF\text{-}CMA}$ security experiment by

$$\mathbf{Adv}^{\mathsf{SIG}}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{EUF\text{-}CMA}^{\mathsf{SIG}}_{\mathcal{A}}(\lambda) = 1].$$

Game $\mathsf{EUF\text{-}CMA}^{\mathsf{SIG}}_{\mathcal{A}}(\lambda)$	Oracle $\textsc{Sign}()$
$\mathsf{par} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$	$\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SIG.KeyGen}(\mathsf{par}); \mathsf{Q} := ()$	$\mathsf{Q} := \mathsf{Q} \| (m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(\mathsf{pk})$	**return** σ
return $(m^* \notin \mathsf{Q} \wedge \mathsf{SIG.Ver}(\mathsf{pk}, m^*, \sigma^*))$	Oracle $\mathsf{Dec}(c)$
	if $c = c^*$ **then return** $\perp$
	return $\mathsf{KEM.Dec}(\mathsf{sk}, c)$

Fig. 1. The EUF-CMA security game for SIG

2.3 Blind Signatures

We follow [34,40] to define a three-move blind signature and its security model.

Definition 4 (Blind Signature). *A three-move blind signature BS with efficient decidable public key space $\mathcal{PK}$ consists of the following PPT algorithms.*

- $\mathsf{BS.Setup}(1^\lambda) \to \mathsf{par}$*: the setup algorithm takes as input the security parameter λ, outputs the public parameter* par*.*
- $\mathsf{BS.KeyGen}(\mathsf{par}) \to (\mathsf{sk}, \mathsf{pk})$*: the key generation algorithm takes as input the public parameter* par*, outputs a pair of public and secret keys* $(\mathsf{sk}, \mathsf{pk})$*.*
- *The signing algorithm* $\mathsf{BS.Sign} = (\mathsf{BS.Sign}_1, \mathsf{BS.Sign}_2)$ *consists of two phases:*
  - $\mathsf{BS.Sign}_1(\mathsf{sk}) \to (\mathsf{state}_S, \rho_{S,1})$*: On input the secret key, outputs an internal signer state* state_S *and the first-sender message* $\rho_{S,1}$*.*
  - $\mathsf{BS.Sign}_2(\mathsf{state}_S, \rho_U) \to \rho_{S,2}$*: On input the signer state* state_S *and a user message* ρ_U*, outputs a second-sender message* $\rho_{S,2}$*.*
- *The user algorithm* $\mathsf{BS.User} = (\mathsf{BS.User}_1, \mathsf{BS.User}_2)$ *consists of two phases:*
  - $\mathsf{BS.User}_1(\mathsf{pk}, M, \rho_{S,1}) \to (\mathsf{state}_U, \rho_U)$*: On input the public key* pk*, a message M and the first-sender message* $\rho_{S,1}$*, outputs an internal user state* state_U *and a user message* ρ_U*.*
  - $\mathsf{BS.User}_2(\mathsf{state}_U, \rho_{S,2})$*: On input a user state* state_U *and a second-signer message* $\rho_{S,2}$*, outputs a signature* σ
- $\mathsf{BS.Ver}(\mathsf{pk}, M, \sigma) \to 0/1$*: On input the public key* pk*, a message M and a signature σ, it outputs* 1 *to indicate the signature is valid, and* 0 *otherwise.*

A blind signature is required to be complete and blind against the malicious signer, and to satisfy one-more unforgeability. Basically, completeness requires that for an honest signer and user, the output signature of a message m should be valid after the interaction. Blindness means that the signer is unable to reveal which sessions it has interacted with the user for generating a signature for a particular message. And briefly speaking, unforgeability requires that after ℓ interactions with the signer for generating signatures, the user cannot forge more than ℓ message-signature pairs.

Definition 5 (Correctness). *A three-move blind signature scheme* BS *is correct if for* $\mathsf{par} \leftarrow \mathsf{BS.Setup}(1^\lambda)$*, all public and secret key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KeyGen}(\mathsf{par})$*, we have*

$$\Pr\left[\mathsf{BS.Ver}(\mathsf{pk}, M, \sigma) = 1 \;\middle|\; \begin{array}{c} (\mathsf{state}_S, \rho_{S,1}) \leftarrow \mathsf{BS.Sign}_1(\mathsf{sk}) \\ (\mathsf{state}_U, \rho_U) \leftarrow \mathsf{BS.User}_1(\mathsf{pk}, M, \rho_{S,1}) \\ \rho_{S,2} \leftarrow \mathsf{BS.Sign}_2(\mathsf{state}_S, \rho_U) \\ \sigma \leftarrow \mathsf{BS.User}_2(\mathsf{state}_U, \rho_{S,2}) \end{array}\right] = 1.$$

Definition 6 (Blindness under Chosen Keys). *For a blind signature* BS*, define the blindness game* BLIND *with an adversary* $\mathcal{A}$ *(playing the signer) as follows.*

Setup. *The challenger samples a bit* $\mathsf{coin} \xleftarrow{\$} \{0,1\}$ *and runs* $\mathcal{A}$ *on input* par*.*

Online Phase. *$\mathcal{A}$ outputs two message M_0^* and M_1^*, a public key pk, the game checks if pk is valid and if so, it assigns $(M_0, M_1) = (M_{\mathsf{coin}}^*, M_{1-\mathsf{coin}}^*)$. If pk is not valid, the game aborts and outputs 0. The adversary $\mathcal{A}$ is given access to oracles $\mathsf{U}_1, \mathsf{U}_2$ which behave as follows.*

- **Oracle $\mathbf{U}_1$.** *On input $b \in \{0,1\}$ and a first-signer message $\rho_{S,1,b}$, if the session b is not yet open, the oracle marks session b as open and generates a state and a challenge as $(\mathsf{state}_{U,b}, \rho_{U,b}) \leftarrow \mathsf{BS.User}_1(\mathsf{pk}, M_b, \rho_{S,1})$. It returns $\rho_{U,b}$ to $\mathcal{A}$.*
- **Oracle $\mathbf{U}_2$.** *On input $b \in \{0,1\}$ and a second-signer message $\rho_{S,2,b}$, if the session b is opened, the oracle creates a signature $\sigma_b \leftarrow \mathsf{BS.User}_2(\mathsf{state}_{U,b}, \rho_{S,2,b})$. It marks session b as closed. Oracle U_2 does not output anything.*

Output Determination. *When both sessions are closed and for $b \in \{0,1\}$ we have that $\mathsf{BS.Ver}(\mathsf{pk}, M_b, \sigma_b) = 1$, the oracle returns the two signatures $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$ to $\mathcal{A}$, where note that σ_{coin} (resp. $\sigma_{1-\mathsf{coin}}$) is a valid signature for M_0^* (resp. M_1^*) regardless of the choice of coin. $\mathcal{A}$ outputs a guess coin^* for coin. We say that $\mathcal{A}$ wins if $\mathsf{coin}^* = \mathsf{coin}$.*

We define the advantage of $\mathcal{A}$ in game $\mathsf{BLIND}_{\mathsf{BS}}$ as

$$\mathbf{Adv}_{\mathsf{BS},\mathcal{A}}^{\mathsf{BLIND}}(\lambda) := \left|\Pr[\mathsf{BLIND} = 1] - \frac{1}{2}\right|.$$

We say that BS is blind under chosen keys if $\mathbf{Adv}_{\mathsf{BS},\mathcal{A}}^{\mathsf{BLIND}}(\lambda)$ is negligible.

We will follow [39] for a generalized definition of one-more unforgeability to cover both sequential (ℓ-SEQ-OMUF) and concurrent (ℓ-OMUF) security notions. Recall that for sequential unforgeability, it is required that the interactions between signer and user need to happen in sequence, i.e., it restricts adversaries to opening no more than one signing session at a time. Now we say that a BS satisfies (ℓ, η)-OMUF if it satisfies ℓ-OMUF and at most η signing sessions can be open concurrently. The one-more unforgeability game is defined in Figure 2. Note that the game checks the maximum number η^* of signing sessions which are open concurrently at any point in time during the game (which should not exceed η).

Game $\mathsf{OMUF}^{\mathcal{A}}_{\mathsf{BS},\ell,\eta}(\lambda)$	Oracle $S_1()$
$\mathsf{par} \leftarrow \mathsf{BS.Setup}(1^\lambda)$	$k_1 \leftarrow k_1 + 1$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KeyGen}(\mathsf{par})$	$(\mathsf{state}_{k_1}, \rho_1) \leftarrow \mathsf{BS.Sign}_1(\mathsf{sk})$
$k_1 \leftarrow 0; k_2 \leftarrow 0; \mathcal{S} \leftarrow \emptyset$	$\mathcal{S} \leftarrow \mathcal{S} \cup \{\|_\infty\}$
$\eta^* \leftarrow 0$	$\eta^* \leftarrow \max(\eta^*, \|\mathcal{S}\|)$
$(m_i^*, \sigma_i^*)_{i=1}^{\ell+1} \leftarrow \mathcal{A}^{S_1, S_2}(pk)$	**return** (k_1, ρ_1)
return $k_2 \leq \ell \wedge \eta^* \leq \eta \wedge$	Oracle $S_2(j, c)$
$\quad \wedge \forall i, j \in [\ell+1] : i \neq j \Rightarrow (m_i^*, \sigma_i^*) \neq (m_j^*, \sigma_j^*)$	**if** $j \notin \mathcal{S}$ **then return** $\perp$
$\quad \wedge \forall i \in [\ell+1] : \mathsf{BS.Ver}(\mathsf{pk}, m_i^*, \sigma_i^*) = 1$	$\rho_2 \leftarrow \mathsf{BS.Sign}_2(\mathsf{sk}, \mathsf{state}_j, c)$
	$\mathcal{S} \leftarrow \mathcal{S} \setminus \{j\}; k_2 \leftarrow k_2 + 1$
	return ρ_2

Fig. 2. The one-more unforgeability game for BS. Here k_1 is the number of signing sessions which have been opened, k_2 is the number of signing sessions which have been closed and should not surpass ℓ, $\mathcal{S}$ is the set of indices of currently open signing sessions, and η^* is the maximum value that $|\mathcal{S}|$ ever takes on, i.e., the maximum number of concurrently opened signing sessions

Definition 7 ((ℓ,η)-One-More Unforgeability ((ℓ,η)-OMUF)). *Let $\ell, \eta \in \mathbb{N}$. We define the (ℓ,η)-OMUF advantage of an adversary $\mathcal{A}$ relative to BS as*

$$\mathbf{Adv}^{\mathit{OMUF}}_{\mathit{BS},\ell,\eta,\mathcal{A}}(\lambda) := \Pr[\mathit{OMUF}^{\mathcal{A}}_{\mathit{BS},\ell,\eta}(\lambda) = 1].$$

We say that the blind signature scheme BS is (ℓ,η)-One-More Unforgeable if $\mathbf{Adv}^{\mathit{OMUF}}_{\mathit{BS},\ell,\eta,\mathcal{A}}(\lambda)$ is negligible in λ.

It is worth noting a useful result from [39] saying that any winning adversary for (ℓ,η)-OMUF can be turned into a winning adversary for $(\ell+\eta,\eta)$-OMUF, instead of $(\ell+\eta,\ell+\eta)$-OMUF as in [34], which closes all of its signing sessions. It is stated formally as follows.

Lemma 1 ([39, Lemma 2]). *Let $\ell, \eta \in \mathbb{N}$ and BS be a blind signature scheme satisfying correctness. Let $\mathcal{A}$ be an adversary for (ℓ,η)-OMUF. Then there exists an adversary $\mathcal{B}$ for $(\ell+\eta,\eta)$-OMUF making exactly $\ell+\eta$ queries to S_1 and exactly $\ell+1$ valid queries to S_2 such that*

$$\mathbf{Adv}^{\mathit{OMUF}}_{\mathit{BS},\ell+\eta,\eta,\mathcal{B}}(\lambda) \geq \mathbf{Adv}^{\mathit{OMUF}}_{\mathit{BS},\ell,\eta,\mathcal{A}}(\lambda).$$

2.4 Key Encapsulation Mechanism (KEM)

Definition 8 (Key Encapsulation Mechanism). *A key encapsulation mechanism scheme KEM consists of the following four algorithms:*

– $\mathsf{KEM.Setup}(1^\lambda) \to \mathsf{par}$: *On input the security parameter* λ*, the setup algorithm outputs public parameter* par*.*
– $\mathsf{KEM.KeyGen}(\mathsf{par}) \to (\mathsf{sk}, \mathsf{pk})$: *On input the public parameter* par*, the key generation algorithm generates a secret key* sk *and a public key* pk*.*
– $\mathsf{KEM.Enc}(\mathsf{pk}) \to (K, c)$: *On input the public key* pk*, the encapsulation algorithm outputs a pair of ciphertext* c *and encapsulated key* $K \in \mathcal{K}$*.*
– $\mathsf{KEM.Dec}(\mathsf{sk}, c) \to K$: *On input the secret key* sk *and a ciphertext* c*, it outputs* $K \in \mathcal{K} \cup \{\perp\}$*.*

Definition 9 (Correctness). *A key encapsulation mechanism scheme* KEM *is correct if for any* λ*, we have that*

$$\Pr\left[\mathsf{KEM.Dec}(\mathsf{sk}, c) = K \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{KEM.Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{par}) \\ (K, c) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}) \end{array}\right] = 1.$$

The strongest security notion for KEM schemes is indistinguishability of ciphertexts under chosen-ciphertext attack (IND-CCA2), where the adversary can query decryptions of encapsulations of its choice even after receiving the challenge. We formalize the IND-CCA2 security game for the KEM scheme in Figure 3.

Game $\mathsf{IND\text{-}CCA2}(\lambda)$	Oracle $\mathsf{Enc}()$ //one time
$\mathsf{par} \leftarrow \mathsf{KEM.Setup}(1^\lambda)$	$(K_0, c^*) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}); K_1 \xleftarrow{\$} \mathcal{K}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{par})$	**return** (K_b, c^*)
$b \xleftarrow{\$} \{0,1\}$	Oracle $\mathsf{Dec}(c)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}(pk)$	**if** $c = c^*$ **then return** $\perp$
return $(b' = b)$	**return** $\mathsf{KEM.Dec}(\mathsf{sk}, c)$

Fig. 3. The IND-CCA2 security game for KEM

Definition 10 (IND-CCA2). *A key encapsulation mechanism scheme* KEM *is IND-CCA2-secure if its advantage in the IND-CCA2 security game in Figure 3, defined by*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \frac{1}{2}\right|,$$

is negligible in λ*.*

3 Cryptographic Group Actions

3.1 Definitions

Let $(\mathcal{G}, \circ)$ be a group with identity id and $\mathcal{X}$ a set. An action of $\mathcal{G}$ on $\mathcal{X}$ is a map $\diamond : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$ satisfying the following properties: (i) $\mathrm{id} \diamond x = x$ for all $x \in \mathcal{X}$

and the identity element $\mathrm{id} \in \mathcal{G}$; and (ii) $g \diamond (h \diamond x) = (g \circ h) \diamond x$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$. A group action is said to be [3]:

- *transitive* if for all $x, y \in \mathcal{X}$, there exists $g \in \mathcal{G}$ such that $g \diamond x = y$;
- *faithful* if there does not exist $g \in \mathcal{G} \setminus \{\mathrm{id}\}$ such that $g \diamond x = x$ for all $x \in \mathcal{X}$, i.e., if $g \diamond x = x$ for all $x \in \mathcal{X}$ then $g = \mathrm{id}$;
- *free* if whenever there exists $x \in \mathcal{X}$ such that $g * x = x$ then $g = id$; and
- *regular* if it is free and transitive.

In shorthand, we write $(\mathcal{G}, \mathcal{X}, \diamond)$ to denote the group action.

Definition 11 (Effective Group Action - EGA). *Let $(\mathcal{G}, \mathcal{X}, \diamond)$ be an effective group action satisfying the following properties:*

1. *The group $\mathcal{G}$ is finite and there exist efficient algorithms for membership testing, equality testing, (random) sampling, group operation and inversion.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $x_0 \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm to evaluate the group action, i.e., to compute $g \diamond x$ given $g \in \mathcal{G}$ and $x \in \mathcal{X}$.*

Then we call $x_0 \in \mathcal{X}$ the origin and $(\mathcal{G}, \mathcal{X}, \diamond, x_0)$ an effective group action (EGA).

In this paper, we restrict our attention to the case of cyclic group actions, i.e., $\mathcal{G}$ being a cyclic group, to capture the isogeny settings ([18]). In addition, we focus on known-order groups ([3]). We will thus focus on *Cyclic Effective Group Action* (CEGA) defined as follows ([3,28]).

Definition 12 (Cyclic Effective Group Action - CEGA). *Let $(\mathcal{G}, \mathcal{X}, \diamond, x_0)$ be an effective group action satisfying the following properties:*

1. *The group $\mathcal{G}$ is cyclic of order N for some known $N \in \mathbb{N}$.*
2. *There exists a generator $g \in \mathcal{G}$, i.e., $\mathcal{G} = \langle g \rangle$, with known representation.*
3. *For any element $h \in \mathcal{G}$, the element $a \in \mathbb{Z}_N$ satisfying $h = g^a$ is efficiently computable.*
4. *The group action $\diamond$ is regular.*

We then say that $(\mathcal{G}, \mathcal{X}, \diamond, x_0)$ is a cyclic (known-order) effective group action (CEGA). We can therefore denote any CEGA equivalent by $(\mathbb{Z}_N, \mathcal{X}, \diamond, x_0)$ where

$$\begin{aligned} \diamond : \mathbb{Z}_N \times \mathcal{X} &\to \mathcal{X} \\ (a, x) &\mapsto \underbrace{(g \circ \cdots \circ g)}_{a \text{ times}} \diamond x. \end{aligned}$$

For our purpose, we introduce CEGA with twists to capture a property of isogeny-based group actions in CSIDH [18].

Definition 13 (Cyclic Effective Group Action with Twists - CEGAT). *Let $(\mathbb{Z}_N, \mathcal{X}, \diamond, x_0)$ be a CEGA. We call it a cyclic effective group action with twists (CEGAT) if there is an efficient algorithm that, given $x = a \diamond x_0$, computes $x^t := -a \diamond x_0$ without knowing $a \in \mathbb{Z}_N$.*

3.2 Isogenies

In this paper, we focus on CSIDH-based group actions in [18], briefly defined as follows (following the description in [11,42,28]).

Let p be a large prime number. Let E denote the elliptic curve over a finite field $\mathbb{F}_p$ and let $\mathbf{0}_E$ be the point at infinity on E. The curve E is called supersingular if and only if the number of $\mathbb{F}_p$-rational points over $\mathbb{F}_p$ is $p+1$, i.e., $\#E(\mathbb{F}_p) = p+1$; otherwise we call the curve an ordinary curve. Note that it can efficiently verify whether an elliptic curve is supersingular by using point counting algorithms such as Schoof's algorithms [52,53]. Rather than counting points, Sutherland [57] provided a more efficient algorithm, relying on structural differences between ordinary and supersingular isogeny graphs, which runs in $O(n^3 \log^2 n)$ time and $O(n)$ space with $n = \log p$.

Given two elliptic curves E and E', an isogeny ϕ is a morphism $\phi : E \to E'$, i.e., it is a map given by rational functions and it is also a group homomorphism such that $\phi(\mathbf{0}_E) = \mathbf{0}_{E'}$. Such two curves are called isogenous. A characterization for this property is given by the group orders: two curves $E/\mathbb{F}_p$ and $E'/\mathbb{F}_p$ are isogenous over $\mathbb{F}_p$ if and only if $\#E(\mathbb{F}_p) = \#E'(\mathbb{F}_p)$. An isomorphism is an isogeny whose inverse over the algebraic closure is also an isogeny and two elliptic curves are isomorphic if and only if they have the same j-invariant, which is a simple algebraic expression in the coefficients of the curve.

A separable isogeny is determined, up to post-composition with an isomorphism, by its kernel, i.e., the set $\ker(\phi) := \{P \in E : \phi(P) = \mathbf{0}_{E'}\}$. Concretely, given a subgroup $G \subseteq E(\mathbb{F}_{p^k})$ of cardinality N, up to post-composition with isomorphisms, there is a unique elliptic curve E' and isogeny $\phi : E \to E'$ of degree N with $\ker(\phi) = G$, i.e., $E' = E/G$. The equation for E' and the isogeny ϕ can be computed using Vélu's formula [59]. For a prime-degree isogeny of degree ℓ, the classical complexity is $\tilde{O}(\ell)$, while later square-root variants reduce this to $\tilde{O}(\sqrt{\ell})$ for large-prime-degree isogenies [9]. Hence, these methods are most practical for relatively small degrees or for degrees that decompose into small prime powers.

The endomorphism $\mathrm{End}(E)$ consists of all isogenies from E to itself and denote by $\mathrm{End}_{\mathbb{F}_p}(E)$ the ring of endomorphisms defined over $\mathbb{F}_p$. For ordinary curve E, one has that $\mathrm{End}(E) = \mathrm{End}_{\mathbb{F}_p}(E)$, but for a supersingular curve E over $\mathbb{F}_p$ we have a strict inclusion $\mathrm{End}_{\mathbb{F}_p}(E) \subsetneqq \mathrm{End}(E)$. It is known that for a supersingular curve E over $\mathbb{F}_p$, its full endomorphism ring $\mathrm{End}(E)$ is an order in a quaternion algebra, whereas $\mathrm{End}_{\mathbb{F}_p}(E)$ is only an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ (cf. [56]).

Fix an elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$. This curve is supersingular and its $\mathbb{F}_p$-rational endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E_0)$ is $\mathcal{O} = \mathbb{Z}[\pi]$ where π is the Frobenius endomorphism $(x, y) \mapsto (x^p, y^p)$. Let $\mathcal{E}ll_p(\mathcal{O})$ be the set of elliptic curves defined over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O}$. The ideal class group $\mathcal{G} := cl(\mathcal{O})$ acts freely and transitively on the set $\mathcal{E}ll_p(\mathcal{O})$. The class group $cl(\mathcal{O})$ was successfully computed for the parameters of CSIDH-512 in [11]. In particular, the authors also proved that $cl(\mathcal{O})$ is a cyclic group generated by $\mathfrak{g} = (3, \pi - 1)$. In

addition, one can identify $cl(\mathcal{O})$ as $\mathbb{Z}_N$ via an efficient computable isomorphism

$$\begin{aligned} \varphi : \mathbb{Z}_N &\to cl(\mathcal{O}) \\ a &\mapsto \mathfrak{g}^a. \end{aligned}$$

Hence, in what follows, we will write an action of $\mathfrak{g}^a$ on an elliptic curve $E \in \mathcal{E}ll_p(\mathcal{O})$, we use only the exponent a and write $a \diamond E$ instead.

Each isomorphism class of a curve with endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$ is represented by a single coefficient $A \in \mathbb{F}_p$, defining the curve

$$E_A : y^2 = x^3 + Ax^2 + x.$$

When $p \equiv 3 \pmod 4$, the quadratic twist of E_A is given by

$$E_A^t : \ -y^2 = x^3 + Ax^2 + x,$$

which is $\mathbb{F}_p$-isomorphic to E_{-A} via the map $(x, y) \mapsto (-x, y)$. Moreover, for any $a \in \mathbb{Z}_N$, we have

$$(a \diamond E)^t = (-a) \diamond E.$$

Therefore, the CSIDH-512 group action can be viewed as a CEGAT. From this point on, we focus on the CSIDH-512 group action. In particular, we identify the group $cl(\mathcal{O})$ with $\mathbb{Z}_N$, denote the set $\mathcal{E}ll_p(\mathcal{O})$ by $\mathcal{X}$, and write E_{-A} for the quadratic twist of E_A. We also slightly abuse notation by sometimes writing E^{-1} for the quadratic twist of E. Naturally, when both subscript and superscript are used, we have $E_{-A}^{-1} = E_A$.

We stress that this twist structure is essential for our construction. More precisely, our Schnorr-style blind signature design requires an efficiently computable involution on action outputs that is compatible with negation in the acting group, and in the CSIDH setting this is exactly what the quadratic twist provides. For group actions without such a structure, the present approach does not apply directly. In particular, we do not claim here a generic instantiation for arbitrary group actions; for settings without a twist-compatible involution, more general frameworks such as [37] are more appropriate.

3.3 Group Action Hardness Assumptions

A group action is *one-way*, if for a random x, the function $f_x : \mathcal{G} \to \mathcal{X}$ defined by $f_x(g) := g \diamond x$ is one-way. The one-way assumption is formulated in [16] as the Group Action Inverse Problem (GAIP), which asks to compute, given $x, y \in \mathcal{X}$, $g \in \mathcal{G}$ such that $g \diamond x = y$, assuming such g exists (which is always the case if the action $\diamond$ is transitive). In this paper, we focus on CEGAT and thereby formalize GAIP as the discrete logarithm problem for group actions ([28]), which we call the *Group Action Discrete Logarithm Problem* defined as follows.

Definition 14 (Group Action Discrete Logarithm Problem - GA-DLOG). *Let CEGAT $= (\mathbb{Z}_N, \mathcal{X}, \diamond, x_0)$ be a cyclic effective group action with twists. We say that an adversary $\mathcal{A}$ solves the group action discrete logarithm problem GA-DLOG if $\mathcal{A}(x_0, a \diamond x_0) = a$ for $a \xleftarrow{\$} \mathbb{Z}_N$.*

We now define the one-more group action discrete logarithm (OM-GA-DLOG) problem that generalizes the one-more discrete logarithm (OMDL) problem in the traditional group-based setting first introduced by Bellare et al. [7]. Our definition follows the style in [40].

Definition 15 (One-More Group Action Discrete Logarithm Problem). *Let $(\mathbb{Z}_N, \mathcal{X}, \diamond, x_0)$ be a CEGAT. For a stateful algorithm $\mathcal{A}$ and a positive integer ℓ, we define the game ℓ-OM-GA-DLOG as follows.*

Setup. *Initialize $C = \emptyset$. Run $\mathcal{A}$ on input x_0.*
Online Phase. *$\mathcal{A}$ is given access to the following oracles:*
- *Oracle chal: takes no input and samples $x \in \mathcal{X}$. It sets $C := C \cup \{x\}$ and returns x.*
- *Oracle gadlog: takes as input two elements x_1, x_2 and returns $a \in \mathbb{Z}_N$ such that $x_1 = a \diamond x_2$[5]. We assume that gadlog can be queried at most ℓ many times.*

Output Determination. *When $\mathcal{A}$ outputs $(x_i, a_i)_{i=1}^{\ell+1}$, return 1 if for all $i \in [\ell+1]$: $x_i \in C$, $x_i = a_i \diamond x_0$ and $a_i \neq a_j$ for $i \neq j$. Otherwise return 0.*

We define the advantage of $\mathcal{A}$ in ℓ-OM-GA-DLOG as

$$\mathbf{Adv}^{\text{OM-GA-DLOG}}_{\ell,\mathcal{A}} := \Pr[\ell\text{-OM-GA-DLOG}^{\mathcal{A}} = 1].$$

Unlike the classical one-more discrete logarithm setting, the reduction in the group-action setting naturally needs the relative action exponent between two action points, not only between a point and a fixed global base. This is because challenge points are generated adaptively by chal(), and the proofs repeatedly need to recover the exponent relating one sampled point to another previously seen point. The 2-input gadlog oracle is therefore the natural relative-action analogue of the classical dlog oracle for our reductions.

3.4 Algebraic Group Action Model

The Algebraic Group Action Model (AGAM) is first defined by Duman et al. in [28], which is analogous to the Algebraic Group Model (AGM) introduced in [33], where the underlying algebraic structure is an effective group instead of a prime-order group. Duman et al. [28] also defined a quantum version for AGAM, though we do not need that version as it is not our focus in this paper.

Definition 16 (Algebraic Group Action Algorithm). *Let CEGA $= (\mathbb{Z}_N, \mathcal{X}, \diamond, x_0)$ be a fixed cyclic effective group action. An algorithm $\mathcal{A}$ is called algebraic, if for each output element $y \in \mathcal{X}$, it additionally provides a representation relative to a previously received set element. Concretely, if $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ is*

[5] We note that our oracle gadlog works differently with the dlog in the classical settings where the input is a group element h and the output is $y \in \mathbb{Z}_p$ such that $h = g^y$ for a fixed group element g.

the list of received set elements so far, $\mathcal{A}$ additionally provides a group element $a \in \mathbb{Z}_N$ and an index $i \in [\ell]$ such that $y = a \diamond x_i$. We denote

$$y_{(i,a)} = a \diamond x_i.$$

If an oracle is queried on some set elements, then $\mathcal{A}$ also has to provide a representation for each set element contained in that query.

Additionally, if the group action is CEGAT, we extend the representation by a bit b indicating whether the base element was twisted before applying the group action, i.e., we denote

$$y_{(i,a,b)} = \begin{cases} a \diamond x_i & \text{if } b = 0, \\ a \diamond x_i^t & \text{if } b = 1. \end{cases} \tag{1}$$

We require that all auxiliary input provided to the adversary that is not in $\mathcal{X}$ does not depend on elements from $\mathcal{X}$.

In this paper, for convention, instead of using the bit $b \in \{0, 1\}$, we use $b \in \{-1, 1\}$ and write $y_{(i,a,b)} = a \diamond x_{bi}$. Note that we have $y_{(i,a,1)} = a \diamond x_i$ for $b = 1$, which is the case $b = 0$ in (1). If $b = -1$, then $y_{(i,a,-1)} = a \diamond x_{-i} = a \diamond x_i^t$, which is the case $b = 1$ in (1).

4 Security of Schnorr Signatures from CEGAT in AGAM

In this section, we prove that the Schnorr signature from CEGAT given in Figure 4 is tightly secure under the hardness of GA-DLOG in the AGAM + ROM. Our proof is inspired by [34, Theorem 1], but the adaptation is not merely formal. Unlike in the classical algebraic group model where the algebraic output can be written as linear combinations with respect to a fixed global generator, the output in the algebraic group action setting is only explained relative to previously seen action points, possibly up to twisting. As a result, the reduction must track the relevant anchor point and action relation, rather than just a coefficient vector. We present the proof in this way because the same flow and technique will be also used in Section 5 for Schnorr blind signatures.

In Figure 4, GrGen is a group generator that, given a security parameter λ, generates a CEGAT $= (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$ as in Section 3.2.

Theorem 1. *Let GrGen be a group generator. Let $\mathcal{A}$ be an algebraic adversary against the EUF-CMA security of the Schnorr signature scheme in Figure 4. running in time at most τ and making at most q_h queries to the random oracle and q_s signature queries. Then there exists an algorithm $\mathcal{B}$ solving GA-DLOG problem with respect to GrGen, running in time at most $\tau + O(q_s + q_h)$, such that*

$$\mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{GA\text{-}DLOG}}_{\mathcal{B}}(\lambda) + \frac{q_s(q_s + q_h) + q_h + 1}{2^\lambda}.$$

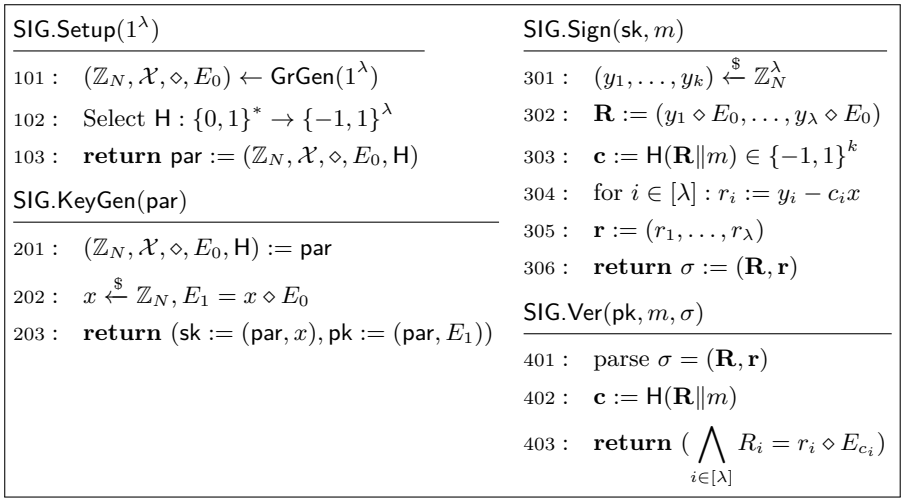

Fig. 4. The Schnorr signature scheme from CEGAT

Let us sketch the idea of the proof first. Assume that the adversary has made q queries $(\mathbf{R}^{(1)}, \mathbf{r}^{(1)}), \ldots, (\mathbf{R}^{(q)}, \mathbf{r}^{(q)})$ and output a forgery $(\mathbf{R}^*, \mathbf{r}^*)$. Since the adversary is algebraic, it also needs to provide $(I, \mathbf{z}, \mathbf{b})$ for some $I = \{(k_i, I_i) : i \in [\lambda], k_i \in [q]\}$, $\mathbf{b} \in \{1, -1\}^\lambda$ such that for $i \in [\lambda]$ either

$$R_i^* = z_i \diamond R_{b_i I_i}^{(k_i)} \quad (= (z_i + b_i r_{I_i}^{(k_i)} + b_i c_{I_i}^{(k_i)} x) \diamond E_0) \tag{2}$$

where $\mathbf{c}^{(k_i)} := H(\mathbf{R}^{(k_i)} \| m^{(k_i)})$; or

$$R_i^* = z_i \diamond E_{b_i} = (z_i + b_i x) \diamond E_0. \tag{3}$$

Since $(\mathbf{R}^*, \mathbf{r}^*)$ is a valid forgery, we have that, for $i \in [\lambda]$:

$$R_i^* = r_i^* \diamond E_{c_i^*} \quad (= (r_i^* + c_i^* x) \diamond E_0) \tag{4}$$

where $\mathbf{c}^* = (c_1^*, \ldots, c_\lambda^*) = H(\mathbf{R}^* \| m^*)$. If (2) happens, then it follows from (2) and (4) that

$$(r_i^* + c_i^* x) \diamond E_0 = (z_i + b_i r_{I_i}^{(k_i)} + b_i c_{I_i}^{(k_i)} x) \diamond E_0. \tag{5}$$

If there exists $i \in [\lambda]$ such that $c_i^* \neq b_i c_{I_i}^{(k_i)}$, then we can extract the secret key x as

$$x := \frac{z_i + b_i r_{I_i}^{(k_i)} - r_i^*}{c_i^* - b_i c_{I_i}^{(k_i)}}.$$

Since c_i^* was chosen at random after the adversary chose b_i and $c_{I_i}^{(k_i)}$, the probability that $\bigwedge_{i \in [\lambda]} c_i^* = b_i c_{I_i}^{(k_i)}$ is negligible.

If (3) happens, then it follows from (3), and (4) that or

$$(r_i^* + c_i^* x) \diamond E_0 = (z_i + b_i x) \diamond E_0. \tag{6}$$

If there exists $i \in [\lambda]$ such that $c_i^* \neq b_i$ then we can extract x as

$$x := \frac{r_i^* - z_i}{b_i - c_i^*}.$$

Since again c_i^* is randomly chosen after adversary provides b_i, the probability that $c_i^* = b_i$ for all $i \in [\lambda]$ is $\frac{1}{2^\lambda}$.

Hence, in both cases, we can extract the secret key x with overwhelming probability. Now we are ready for the formal proof.

Proof of Theorem 1. Let $\mathcal{A}$ be an algebraic adversary in the EUF-CMA game that makes at most q_S signature queries and q_h random oracle queries. We proceed by a sequence of games defined in Figure 5. The proof follows the same flow as in the proof of [34, Theorem 1]. For convenience, we set $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_i}(\lambda) := \Pr[\mathsf{Game}_i^{\mathcal{A}} = 1]$.

Game_0. This first game is EUF-CMA game for the Schnorr signature scheme in Figure 4 with a random oracle H. It maintains a list of queries Q and a list T of values sampled for H. Since the adversary is algebraic, it has to provide $(I, \mathbf{z}, \mathbf{b})$ together with the forgery $(\mathbf{R}^*, \mathbf{r}^*)$ for the message m^* such that either $R_i^* = z_i \diamond E_{b_i}$ or $R_i^* = z_i \diamond R_{b_i I_i}^{(k_i)}$ for $i \in [\lambda]$, and similarly for each oracle query $\mathsf{H}(\mathbf{R}_{(I,\mathbf{z},\mathbf{b})}, m)$. We make a change so that it is necessary to first check $m^* \notin \mathsf{Q}$ and then run the signature verification on $(m^*, (\mathbf{R}^*, \mathbf{r}^*))$. Moreover, we force the game to call $\mathbf{c}^* := \mathsf{H}(\mathbf{R}^*_{(I,\mathbf{z},\mathbf{b})} \| m^*)$ instead of calling $\mathbf{c}^* := \tilde{\mathsf{H}}(\mathbf{R}^* \| m^*)$. However, in Game_0, both oracles are the same. Thus, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0}(\lambda) = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda). \tag{7}$$

Game_1. In this game, when $\mathsf{H}(\mathbf{R}_{(I,\mathbf{z},\mathbf{b})} \| m)$ is called, it will return 0 whenever the freshly random $\mathbf{t} := \mathsf{T}(\mathbf{R}, m) \xleftarrow{\$} \{-1, 1\}^\lambda$ satisfies $t_i = b_i$ or $t_i = b_i c_{I_i}^{(k_i)}$ for $i \in [\lambda]$. As shown above, this happens with probability $\frac{1}{2^\lambda}$. Since H is called at most q_h times by the adversary, and once by the game when it checks the validity of the signature, by union bound, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0}(\lambda) - \frac{q_h + 1}{2^\lambda}. \tag{8}$$

Game_2. In this game, we answer the signature query by simulating the oracle SIGN without the secret key x by programming the random oracle. We see that Game_1 and Game_2 are the same unless Game_2 returns 0 in (II). For each signature query, $\mathbf{R}^{(j)}$ is uniformly random, and the size of the table $\mathbf{T}$ is at most $q_h + q_s$, hence the game aborts in line 504 with probability at most $(q_h + q_s)/2^\lambda$. By summing up all q_s signature queries, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}(\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda) - \frac{q_s(q_h + q_s)}{2^\lambda}. \tag{9}$$

Game_0, $\boxed{\mathsf{Game}_1}$, $\boxed{\mathsf{Game}_2}$

101 : $(\mathbb{Z}_N, \mathcal{X}, E_0) \leftarrow \mathsf{GrGen}(1^\lambda)$

102 : $j := 0; x \xleftarrow{\$} \mathbb{Z}_N; E_1 = x \diamond E_0$

103 : $\mathsf{Q} := (); \mathsf{T} := ();$

104 : $(m^*, (\mathbf{R}^*_{(I,\mathbf{z},\mathbf{b})}, \mathbf{r}^*)) \leftarrow \mathcal{A}^{\mathsf{H},\textsc{Sign}}(\mathbb{Z}_N, \mathcal{X}, E_0)$

105 : if $m^* \in \mathsf{Q}$ then return 0

106 : $\mathbf{c}^* = \mathsf{H}(\mathbf{R}^* \| m^*)$

107 : **return** $\bigwedge_{i=1}^{\lambda} (R_i^* = r_i^* \diamond E_{c_i^*})$

Oracle $\mathsf{H}(\mathbf{R}_{(I,\mathbf{z},\mathbf{b})} \| m)$

201 : // $R_i = z_i \diamond E_{b_i}$ or $z_i \diamond R^{(k_i)}_{b_i I_i}$ $\forall\ i \in [\lambda]$

202 : if $\mathsf{T}(\mathbf{R}, m) \perp$ then

203 : $\mathbf{t} := \mathsf{T}(\mathbf{R}, m) \xleftarrow{\$} \{-1, 1\}^\lambda$

204 : $\boxed{\text{if } t_i = b_i \text{ or } b_i c_{I_i}^{(k_i)} \ \forall\ i \in [\lambda] \text{ abort and return } 0}$

205 : **return** $\mathsf{T}(\mathbf{R}, m)$

Oracle $\tilde{\mathsf{H}}(\mathbf{R} \| m)$

301 : if $\mathsf{T}(\mathbf{R}, m) \perp$ then

302 : $\mathsf{T}(\mathbf{R}, m) \xleftarrow{\$} \{-1, 1\}^\lambda$

303 : **return** $\mathsf{T}(\mathbf{R}, m)$

Oracle $\textsc{Sign}(m)$

401 : $j := j + 1; \mathbf{y} \xleftarrow{\$} \mathbb{Z}_N^\lambda;$

402 : $R_i^{(j)} := y_i \diamond E_0\ \forall i \in [\lambda]$

403 : $\mathbf{c}^{(j)} := \tilde{\mathsf{H}}(\mathbf{R}^{(j)} \| m);$

404 : $r_i^{(j)} := y_i - c_i^{(j)} x\ \forall i \in [\lambda]$

405 : $\mathsf{Q} := \mathsf{Q} \| (m)$

406 : **return** $(\mathbf{R}^{(j)}, \mathbf{r}^{(j)})$

Oracle $\textsc{Sign}(m)$

501 : $j := j + 1; \mathbf{r} \xleftarrow{\$} \mathbb{Z}_N^\lambda; \mathbf{c} \xleftarrow{\$} \{-1, 1\}^\lambda$

502 : $R_i^{(j)} := r_i \diamond E_{c_i}\ \forall i \in [\lambda]$

503 : if $\mathsf{T}(\mathbf{R}^{(j)}, m) = \perp$ then $\mathsf{T}(\mathbf{R}^{(j)}, m) := \mathbf{c}$

504 : else abort game and return 0

505 : $\mathsf{Q} := \mathsf{Q} \| (m)$

506 : **return** $(\mathbf{R}^{(j)}, \mathbf{r}^{(j)})$

Fig. 5. Games in the proof of Theorem 1; Game 0 ignores all boxes.

Reduction to GA-DLOG. We now construct an algorithm $\mathcal{B}$ that solves GA-DLOG with the same probability that the adversary $\mathcal{A}$ wins Game_2 as follows. On input the public parameter $(\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$ and E_1, $\mathcal{B}$ runs $\mathcal{A}$ on $(\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, E_1)$ and simulates Game_0. Assume that the adversary $\mathcal{A}$ wins Game_2 and returns $(m^*, (\mathbf{R}^*, \mathbf{r}^*)$ together with $(I, \mathbf{z}, \mathbf{b})$ with $\mathbf{c}^* := \mathsf{T}(\mathbf{R}^*, m^*)$ which is defined as in Game_2 after $\mathcal{A}$ called $\mathbf{c}^* := \mathsf{H}(\mathbf{R}^*_{(I,\mathbf{z},\mathbf{b})} \| m^*)$.

Note that $m^* \notin \mathsf{Q}$, since otherwise it means that the game has returned 0. Therefore, $\mathsf{T}(\mathbf{R}^*, m^*)$ can only be defined either during a call to H by the adversary or by the game when calling $\mathbf{c}^* := \mathsf{H}(\mathbf{R}^*_{(I,\mathbf{z},\mathbf{b})} \| m^*)$ if it is still undefined when $\mathcal{A}$ stops. In both cases, it sets both $\mathsf{T}(\mathbf{R}^*, m^*)$ and $(I, \mathbf{z}, \mathbf{b})$, and hence $(I, \mathbf{z}, \mathbf{b}) \neq \perp$. Moreover, $c_i^* \neq b_i c_{I_i}^{(k_i)}$ for some $i \in [\lambda]$ since otherwise the game will abort and return 0 as in line 204.

Since either $R_i^* = z_i \diamond E_{b_i}$ or $R_i^* = z_i \diamond R_{b_i I_i}^{(k_i)} = (z_i + b_i r_{I_i}^{(k_i)} + b_i c_{I_i}^{(k_i)} x) \diamond E_0$ for $i \in [\lambda]$ and by the validity of the forgery $R_i^* = r_i \diamond E_{c_i^*} = (r_i^* + c_i^* x) \diamond E_0$, it follows that either

$$(z_i + b_i x) \diamond E_0 = (r_i^* + c_i^* x) \diamond E_0 \text{ or } (z_i + b_i r_{I_i}^{(k_i)} + b_i c_{I_i}^{(k_i)} x) \diamond E_0 = (r_i^* + c_i^* x) \diamond E_0$$

which implies that $\mathcal{B}$ can compute the secret key x as either

$$x := \frac{z_i - r_i^*}{c_i^* - b_i} \text{ or } x := \frac{z_i + b_i r_{I_i}^{(k_i)} - r_i^*}{c_i^* - b_i c_{I_i}^{(k_i)}}.$$

Therefore

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{GA\text{-}DLOG}}(\lambda) = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}(\lambda). \tag{10}$$

It follows from Equations (7)-(10) that

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{GA\text{-}DLOG}}(\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda) - \frac{q_s(q_s + q_h) + q_h + 1}{2^\lambda}.$$

Assuming that the multiplications in $\mathbb{Z}$ and assignments in table T take unit time, the running time of $\mathcal{B}$ is $\tau + O(q_s + q_h)$. □

5 Schnorr Blind Signatures from CEGAT in AGAM

In this section, we describe the Schnorr blind signature from CEGAT. Note that this scheme was mentioned by Katsumata et al in [42]; however, there exists no proof yet. The description of the scheme consists of the following algorithms.

- $\mathsf{BS.Setup}(1^\lambda)$: On input the security parameter, it invokes the group generator GrGen as in Section 4 to generate a $\mathsf{CEGAT} = (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$ as a public parameter par.
- $\mathsf{BS.KeyGen}(\mathsf{par})$: On input the parameter $\mathsf{par} = (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$, samples $x \xleftarrow{\$} \mathbb{Z}_N$ and computes $E_1 := x \diamond E_0$. It sets $\mathsf{sk} := x$ and $\mathsf{pk} := E_1$ and returns $(\mathsf{sk}, \mathsf{pk})$.

- $\mathsf{BS.Sign}_1(\mathsf{sk})$: On input the secret key $\mathsf{sk} = x$, it samples $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_N^\lambda$ and returns a commitment $\mathbf{Y} := \mathbf{y} \diamond E_0 = (y_1 \diamond E_0, \ldots, y_\lambda \diamond E_0)$ as the first-sender message and the state $\mathsf{state}_S = \mathbf{y}$.
- $\mathsf{BS.Sign}_2(\mathsf{state}_S, \mathbf{c})$: On input the secret key $\mathsf{sk} = x$, the state $\mathsf{state}_S = \mathbf{y}$ and a challenge $\mathbf{c} \in \{-1, 1\}^\lambda$, computes $\mathbf{r} = \mathbf{y} - \mathbf{c} \cdot x = (y_1 - c_1 x, \ldots, y_\lambda - c_\lambda x)$ and returns the response $\mathbf{r}$ as the second-sender message.
- $\mathsf{BS.User}_1(\mathsf{pk}, m, \mathbf{Y})$: On input the public key $\mathsf{pk} := E_1$, a message M, and the commitment $\mathbf{Y}$ received from $\mathsf{BS.Sign}_1$, it samples $(\mathbf{d}, \mathbf{w}) \xleftarrow{\$} \{-1, 1\}^\lambda \times \mathbb{Z}_N^\lambda$, computes $\mathbf{Y}' = \mathbf{w} \diamond \mathbf{Y}^{\mathbf{d}} = (w_1 \diamond Y_1^{d_1}, \ldots, w_\lambda \diamond Y_\lambda^{d_\lambda})$. It then computes $\mathbf{c}' = \mathsf{H}(\mathbf{Y}' \| m)$ and $\mathbf{c} = \mathbf{d} \cdot \mathbf{c}' = (d_1 c_1, \ldots, d_\lambda c_\lambda)$. It returns the challenge $\mathbf{c}$ and the state $\mathsf{state}_U = (\mathbf{Y}, \mathbf{c}, \mathbf{d}, \mathbf{w}, m)$.
- $\mathsf{BS.User}_2(\mathsf{state}_U, \mathbf{r})$: On input the state $\mathsf{state}_U = (\mathbf{Y}, \mathbf{c}, \mathbf{d}, \mathbf{w}, m)$ and a response $\mathbf{r}$, it checks if $\mathbf{r} \diamond E_1^{\mathbf{c}} = \mathbf{Y}$, i.e., check for every $i \in [\lambda]$ if $r_i \diamond E_1^{c_i} = Y_i$. If yes then it returns a signature $\sigma := (\mathbf{Y}', \mathbf{d} \cdot \mathbf{r} + \mathbf{w})$, otherwise it returns $\perp$.
- $\mathsf{BS.Ver}(\mathsf{pk}, m, \sigma)$: On input the public key $\mathsf{pk} = E_1$, a message m and a signature $\sigma = (\mathbf{Y}', \mathbf{r}')$, let $\mathbf{c}' = \mathsf{H}(\mathbf{Y}', m)$. If $\mathbf{r}' \diamond E_1^{\mathbf{c}'}$, i.e., $r_i' \diamond E_1^{c_i'} = Y_i'$ for $i \in [\lambda]$, return 1; otherwise return 0.

The Schnorr blind signature BS is depicted in Figure 6.

Note that the Schnorr blind signature BS is obtained from the Schnorr signature SIG in Figure 4 by replacing the $\mathsf{SIG.Sign}$ with the interactive protocol specified in Figure 7. The correctness of BS follows since, for $\mathbf{c}' = \mathsf{H}(\mathbf{Y}' \| m)$, if $\mathbf{r} \diamond E_1^{\mathbf{c}} = \mathbf{Y}$ is satisfied in Line 601 in Figure 6, then

$$\mathbf{r}' \diamond E_1^{\mathbf{c}'} = (\mathbf{d} \cdot \mathbf{r} + \mathbf{w}) \diamond E_1^{\mathbf{c} \cdot \mathbf{d}} = \mathbf{w} \diamond \mathbf{Y}^{\mathbf{d}} = \mathbf{Y}'$$

where the last equation follows from Line 402 in Figure 6. For honest signer and user, the condition $\mathbf{r} \diamond E_1^{\mathbf{c}} = \mathbf{Y}$ in Line 601 in Figure 6 is always satisfied since

$$\mathbf{r} \diamond E_1^{\mathbf{c}} = (\mathbf{y} - \mathbf{c} \cdot x) \diamond E_1^{\mathbf{c}} = \mathbf{y} \diamond E_0 = \mathbf{Y}. \tag{11}$$

In fact, this is the correctness condition of the underlying signature SIG.

5.1 Blindness

The proof of blindness is standard. Note that the original Schnorr blind signature achieves perfect blindness [22]. We also prove the perfect blindness for our BS in this paper. Since checking E_1 is a valid elliptic curve (i.e., supersingular) can be done efficiently and for such a valid E_1, there exists a unique $x \in \mathbb{Z}_N$ such that $x \diamond E_0 = E_1$, our blind signature BS is secure even against a malicious signer outputting an arbitrary public key, as noted in [42] for CSI-Otter.

Theorem 2 (Blindness). *The Schnorr blind signature BS in Figure 6 is perfectly blind under chosen keys.*

Proof. We follow the standard approach of [41,42,29]. It is enough to show that for any valid public key pk, any accepting transcript

$$\tau = (\mathbf{Y}, \mathbf{c}, \mathbf{r}),$$

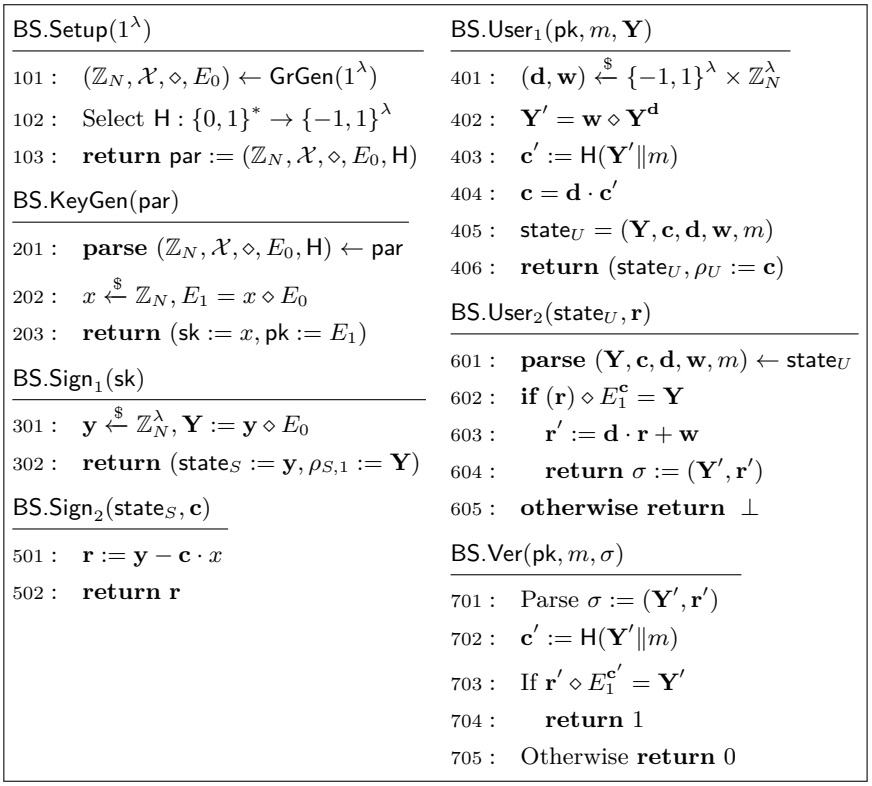

Fig. 6. The Schnorr blind signature scheme BS from CEGAT

where $\mathbf{Y}$ is the first signer message, $\mathbf{c}$ is the user challenge, and $\mathbf{r} \in \mathbb{Z}_N^\lambda$ is the second signer message satisfying

$$\mathbf{r} \diamond E_1^{\mathbf{c}} = \mathbf{Y},$$

and any valid signature $\sigma = (\mathbf{Y}', \mathbf{r}')$ on message m, there exists a unique user randomness $(\mathbf{d}, \mathbf{w})$ (equivalently, a unique user state $\mathsf{state}_U = (\mathbf{Y}, \mathbf{c}, \mathbf{d}, \mathbf{w}, m)$) that could have generated σ.

Note that any public key $\mathsf{pk} = E_1$ output by the adversary (i.e., a malicious signer) can be efficiently checked to be a valid elliptic curve. For such a valid E_1, there exists a unique secret key $x \in \mathbb{Z}_N$ such that $E_1 = x \diamond E_0$.

Now fix $\mathsf{pk} = E_1$, an accepting transcript $\tau = (\mathbf{Y}, \mathbf{c}, \mathbf{r})$, and a valid signature $\sigma = (\mathbf{Y}', \mathbf{r}')$ on m. Let

$$\mathbf{c}' := \mathsf{H}(\mathbf{Y}' \| m) \in \{-1,1\}^\lambda,$$

and define

$$\mathbf{d} := \mathbf{c} \cdot \mathbf{c}' \in \{-1,1\}^\lambda, \qquad \mathbf{w} := \mathbf{r}' - \mathbf{d} \cdot \mathbf{r} \in \mathbb{Z}_N^\lambda,$$

$\mathsf{BS.Sign}((\mathbb{Z}_N, \mathcal{X}, \diamond, E_0), x)$		$\mathsf{BS.User}((\mathbb{Z}_N, \mathcal{X}, \diamond, E_0), E_1)$
$\mathbf{y} \xleftarrow{\$} \mathbb{Z}_N^\lambda$		
$\mathbf{Y} = (y_1 \diamond E_0, \ldots, y_\lambda \diamond E_0)$	$\xrightarrow{\mathbf{Y}}$	$(\mathbf{d}, \mathbf{w}) \xleftarrow{\$} \{-1, 1\}^\lambda \times \mathbb{Z}_N^\lambda$
		$\mathbf{Y}' = \mathbf{w} \diamond \mathbf{Y}^{\mathbf{d}}$
		$\mathbf{c}' = \mathsf{H}(\mathbf{Y}' \Vert m)$
	$\xleftarrow{\mathbf{c}}$	$\mathbf{c} = \mathbf{c}' \cdot \mathbf{d}$
$\mathbf{r} = \mathbf{y} - \mathbf{c} \cdot x$	$\xrightarrow{\mathbf{r}}$	if $\mathbf{r} \diamond E_1^{\mathbf{c}} \neq \mathbf{Y}$ return $\perp$
		$\mathbf{r}' := \mathbf{d} \cdot \mathbf{r} + \mathbf{w}$
return 1		**return** $\sigma := (\mathbf{Y}', \mathbf{r}')$

Fig. 7. The signing interaction between signer and user in BS

where all products are component-wise. Since $\mathbf{c}, \mathbf{c}' \in \{-1, 1\}^\lambda$, we also have

$$\mathbf{c}' = \mathbf{c} \cdot \mathbf{d}.$$

Because the transcript is accepting, Equation (11) gives

$$\mathbf{Y}^{\mathbf{d}} = (\mathbf{d} \cdot \mathbf{r}) \diamond E_1^{\mathbf{c} \cdot \mathbf{d}} = (\mathbf{d} \cdot \mathbf{r}) \diamond E_1^{\mathbf{c}'}.$$

Since $\sigma = (\mathbf{Y}', \mathbf{r}')$ is a valid signature, we have

$$\mathbf{Y}' = \mathbf{r}' \diamond E_1^{\mathbf{c}'}.$$

Substituting $\mathbf{r}' = \mathbf{d} \cdot \mathbf{r} + \mathbf{w}$, we obtain

$$\mathbf{Y}' = (\mathbf{w} + \mathbf{d} \cdot \mathbf{r}) \diamond E_1^{\mathbf{c}'} = \mathbf{w} \diamond \left((\mathbf{d} \cdot \mathbf{r}) \diamond E_1^{\mathbf{c}'} \right) = \mathbf{w} \diamond \mathbf{Y}^{\mathbf{d}}.$$

Hence $(\mathbf{d}, \mathbf{w})$ is indeed a valid choice of user randomness that would produce the blinded commitment $\mathbf{Y}'$ in Figure 6. Moreover, with this choice of $(\mathbf{d}, \mathbf{w})$, the user computes

$$\mathbf{r}' = \mathbf{d} \cdot \mathbf{r} + \mathbf{w},$$

so the final output is exactly $\sigma = (\mathbf{Y}', \mathbf{r}')$.

We now show uniqueness. Suppose that the same accepting transcript $\tau = (\mathbf{Y}, \mathbf{c}, \mathbf{r})$ and the same valid signature $\sigma = (\mathbf{Y}', \mathbf{r}')$ can be produced from two user randomness pairs $(\mathbf{d}_1, \mathbf{w}_1)$ and $(\mathbf{d}_2, \mathbf{w}_2)$. Since σ is fixed, the value

$$\mathbf{c}' = \mathsf{H}(\mathbf{Y}' \Vert m)$$

is fixed. Since $\mathbf{c}$ is also fixed by the transcript, we must have

$$\mathbf{d}_1 = \mathbf{c} \cdot \mathbf{c}' = \mathbf{d}_2.$$

Then from

$$\mathbf{r}' = \mathbf{d}_1 \cdot \mathbf{r} + \mathbf{w}_1 = \mathbf{d}_2 \cdot \mathbf{r} + \mathbf{w}_2$$

it follows that $\mathbf{w}_1 = \mathbf{w}_2$. Therefore the user randomness $(\mathbf{d}, \mathbf{w})$ is unique.

Thus, for every accepting transcript and every valid output signature, there exists a unique user randomness, equivalently a unique user state, that could have generated it. This gives a bijection between valid signatures and user randomness for any fixed signer view, which proves perfect blindness under chosen keys. In particular, for any accepting transcript $(\mathbf{Y}, \mathbf{c}, \mathbf{r})$ and any valid signature $(\mathbf{Y}', \mathbf{r}')$, the values $\mathbf{d} = \mathbf{c} \cdot \mathsf{H}(\mathbf{Y}' \| m)$ and $\mathbf{w} = \mathbf{r}' - \mathbf{d} \cdot \mathbf{r}$ always satisfy $\mathbf{Y}' = \mathbf{w} \diamond \mathbf{Y}^{\mathbf{d}}$, so this relation cannot be used to link a signature to a unique signer transcript. □

5.2 Sequential One-More Unforgeability

The proof of sequential one-more unforgeability of our Schnorr blind signature scheme BS in Figure 6 in AGAM + ROM follows the framework of the proof of the classical Schnorr blind signature in AGM + ROM by Kastner, Loss, and Xu [40]. The main result is stated as follows.

Theorem 3. *Let $\mathcal{A}$ be an algebraic adversary that runs in time τ, makes at most ℓ queries to $\mathsf{BS.Sign}_2$, and at most q_h random oracle queries to H. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{OM\text{-}GA\text{-}DLOG}}_{\mathcal{B}}(\lambda) \geq \mathbf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\ell,1,\mathcal{A}}(\lambda) - \frac{q_h^2 + 3q_h + 2}{2^{\lambda+1}}.$$

Let us sketch the idea of the proof first. The idea and notation are reused from [40,34], which we include for completeness. Since $\mathcal{A}$ can query $\mathsf{BS.Sign}_2$ a total of ℓ times, it is allowed to open $\ell+1$ sessions and close the first ℓ of them (the last session is never closed). Let $\mathbf{Y}^{(1)}, \cdots, \mathbf{Y}^{(\ell+1)}$ be the commitments returned by $\mathsf{BS.Sign}_1$. Let $(m_1, (\mathbf{R}^{(1)}, \mathbf{r}^{(1)})), \cdots, (m_{\ell+1}, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)}))$ be the output of $\mathcal{A}$, i.e., $(\mathbf{R}^{(i)}, \mathbf{r}^{(i)})$ is the forgery of m_i for $i \in [\ell+1]$. Since $\mathcal{A}$ is algebraic, for each $i \in [\ell+1]$, it also needs to output $(I^{(i)}, \mathbf{z}^{(i)}, \mathbf{b}^{(i)})$ with $I^{(i)} = \{(k_j^{(i)}, I_j^{(i)}) \in [\ell+1] \times [\lambda] : j \in [\lambda]\}$, $\mathbf{b}^{(i)} \in \{1, -1\}^\lambda$ such that for $j \in [\lambda]$ either

$$R_j^{(i)} = z_j^{(i)} \diamond E_1^{b_j^{(i)}} \tag{12}$$

or

$$R_j^{(i)} = z_j^{(i)} \diamond Y_{b_j^{(i)} I_j^{(i)}}^{(k_j^{(i)})}. \tag{13}$$

Since $(\mathbf{R}^{(i)}, \mathbf{r}^{(i)})$ is the forgery of m_i for $i \in [\ell+1]$, we have for $j \in [\lambda]$ that:

$$R_j^{(i)} = r_j^{(i)} \diamond E_1^{c_j^{*(i)}} \tag{14}$$

where $\mathbf{c}^{*(i)} := \mathsf{H}(\mathbf{R}^{(i)} \| m_i)$. The above three equations can be combined to yield either

$$r_j^{(i)} \diamond E_1^{c_j^{*(i)}} = z_j^{(i)} \diamond E_1^{b_j^{(i)}} \tag{15}$$

or

$$r_j^{(i)} \diamond E_1^{c_j^{*(i)}} = z_j^{(i)} \diamond Y_{b_j^{(i)} I_j^{(i)}}^{(k_j^{(i)})} \tag{16}$$

for $i \in [\ell + 1], j \in [\lambda]$.

The reduction to OM-GA-DLOG $\mathcal{B}$ works as follows. $\mathcal{B}$ queries its challenge oracle $\ell\lambda + 1$ times to obtain $E_1, \mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(\ell)}$, samples $\mathbf{y}^{(\ell+1)} \xleftarrow{\$} \mathbb{Z}_N^\lambda$ and sets $\mathbf{Y}^{(\ell+1)} := \mathbf{y}^{(\ell+1)} \diamond E_0$, and simulates $\mathsf{BS.Sign}_1$ by returning $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(\ell)}, \mathbf{Y}^{(\ell+1)}$. To simulate $\mathsf{BS.Sign}_2(\mathbf{c}^{(i)})$ query, $\mathcal{B}$ queries its gadlog oracle λ times and returns $\mathbf{r}^{(i)}$ such that $\mathbf{r}^{(i)} \diamond E_1^{\mathbf{c}^{(i)}} = \mathbf{Y}^{(i)}$. If (16) happens, substituting this into Equation (16), we get

$$r_j^{(i)} \diamond E_1^{c_j^{*(i)}} = (z_j^{(i)} + b_j^{(i)} r_{I_j^{(i)}}^{(k_j^{(i)})}) \diamond E_1^{b_j^{(i)} c_j^{(k_j^{(i)})}} \tag{17}$$

which can be used to compute the secret x as

$$x := \frac{z_j^{(i)} + b_j^{(i)} r_{I_j^{(i)}}^{(k_j^{(i)})} - r_j^{(i)}}{c_j^{*(i)} - b_j^{(i)} c_j^{(k_j^{(i)})}} \tag{18}$$

as long as $c_j^{*(i)} \neq b_j^{(i)} c_j^{(k_j^{(i)})}$ for some $i \in [\ell + 1]$ and $j \in [\lambda]$. We will show that the probability of the event $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})}$ for all $i \in [\ell + 1]$ and all $j \in [\lambda]$ is negligible. If (15) happens, then we can obtain the secret x as

$$x := \frac{z_j^{(i)} - r_j^{(i)}}{c_j^{*(i)} - b_j^{(i)}}$$

as long as $c_j^{*(i)} \neq b_j^{(i)}$ for some $i \in [\ell + 1]$ and $j \in [\lambda]$. The probability of the event $c_j^{*(i)} = b_j^{(i)}$ for all $i \in [\ell + 1]$ and all $j \in [\lambda]$ is negligible. Hence, in both cases, we can extract the secret key with overwhelming probability. We're now ready for the formal proof.

Proof of Theorem 3. The proof proceeds in a similar manner as that of [40, Theorem 2]. Let $\mathcal{A}$ be an algebraic adversary in the OMUF game that makes at most ℓ queries to $\mathsf{BS.Sign}_2$ and at most q_h queries to the random oracle H. Without loss of generality, we assume that $\mathcal{A}$ makes exactly $\ell + 1$ queries to $\mathsf{BS.Sign}_1$ and exactly ℓ queries to $\mathsf{BS.Sign}_2$, and returns exactly $\ell + 1$ forgeries $(\mathbf{R}^{(1)}, \mathbf{r}^{(1)}), \ldots, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)})$ for messages $m_1, \ldots, m_{\ell+1}$.

First note that the pairs $(m_1, \mathbf{R}^{(1)}), \ldots, (m_{\ell+1}, \mathbf{R}^{(\ell+1)})$ are all distinct. In fact, suppose that $(m_i, \mathbf{R}^{(i)}) = (m_j, \mathbf{R}^{(j)})$ for $i \neq j$. If $\mathbf{r}^{(i)} = \mathbf{r}^{(j)}$, then $\mathcal{A}$ outputs two identical message-signature pairs, which violates the winning condition. Otherwise, it cannot be the case that both $(\mathbf{R}^{(i)}, \mathbf{r}^{(i)})$ and $(\mathbf{R}^{(j)}, \mathbf{r}^{(j)})$ are valid signatures of m_i, since given m_i and $\mathbf{R}^{(i)}$, the signature $\mathbf{r}^{(i)}$ is uniquely defined as in Equation (14).

Now we let E_1 be the public key, $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(\ell+1)}$ be the commitments returned by $\mathsf{BS.Sign}_1$ and let $\mathsf{BS.Sign}_2(\mathbf{c}^{(1)}), \ldots, \mathsf{BS.Sign}_2(\mathbf{c}^{(\ell)})$ be the queries of $\mathcal{A}$ to $\mathsf{BS.Sign}_2$. The proof proceeds by a sequence of games as follows. For convenience, we set $\mathbf{Adv}^{\mathsf{Game}_i}_{\mathcal{A}}(\lambda) := \Pr[\mathsf{Game}_i^{\mathcal{A}} = 1]$ for $i = \{0, 1\}$.

Game_0. This is the original game that captures sequential security, and thus,

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\ell,1,\mathcal{A}}(\lambda).$$

Game_1. In Game_1, we make the following change. When $\mathcal{A}$ returns its final outputs $(m_1, (\mathbf{R}^{(1)}, \mathbf{r}^{(1)})), \ldots, (m_{\ell+1}, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)}))$ together with $(I^{(i)}, \mathbf{z}^{(i)}, \mathbf{b}^{(i)})$ for $i \in [\ell + 1]$, we emulate a query $\mathbf{c}^{*(i)} := \mathsf{H}(\mathbf{R}^{(i)} \| m_i)$ via lazy sampling. We will abort if either $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})}$ or $c_j^{*(i)} = b_j^{(i)}$ for $i \in [\ell + 1]$ and $j \in [\lambda]$. We denote by E this event, and we will first bound $\Pr[E]$.

If $\mathcal{A}$ does not query $\mathsf{H}(\mathbf{R}^{(i)} \| m_i)$ for some i, then $\mathbf{c}^{*(i)}$ is uniformly random element of $\{-1, 1\}^\lambda$ from the adversary's view, and

$$\Pr[c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})} \vee c_j^{*(i)} = b_j^{(i)}] = \frac{1}{2^\lambda}.$$

Now assume that $\mathcal{A}$ queries $\mathsf{H}(\mathbf{R}^{(i)} \| m_i)$ for all i; we follow [40] to call such query *i-th special query*. Since $(m_i, \mathbf{R}^{(i)})$ pairs are all distinct, $\mathbf{c}^{*(i)} := \mathsf{H}(\mathbf{R}^{(i)} \| m_i)$ is uniformly random when $\mathcal{A}$ makes the i-th special query. In particular, the tuple $(I^{(i)}, \mathbf{z}^{(i)}, \mathbf{b}^{(i)})$ is already determined when $\mathcal{A}$ makes the i-th special query. Since we are in the sequential setting, where the adversary must close one session before opening another, any special query is made either during a session which is eventually closed (i.e., between $\mathcal{A}$'s j-th $\mathsf{BS.Sign}_1$ query and j-th $\mathsf{BS.Sign}_2$ query for some $j \in [\ell]$), or between two sessions (including before the first session), or during the last session which is never closed (i.e., after $\mathcal{A}$'s $(\ell+1)$-th $\mathsf{BS.Sign}_1$ query). We consider the following cases.

- Case E_1: There is any special query (say the i-th) made (a) between two sessions including before the first session, i.e., made after the j_0-th $\mathsf{BS.Sign}_2$ query and before the $(j_0 + 1)$-th $\mathsf{BS.Sign}_1$ query, or (b) after the $(\ell + 1)$-th $\mathsf{BS.Sign}_1$ query. We consider the time when $\mathcal{A}$ makes the i-th special query $\mathsf{H}(\mathbf{R}^{(i)} \| m_i)$.
  - In case (a), at this point, $\mathcal{A}$ has already seen $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(j_0)}$, and $b_j^{(i)} c_j^{(k_j^{(i)})}$'s are fixed, for $k_j^{(i)} \leq j_0$, $j \in [\lambda]$, when $\mathcal{A}$ makes the $k_j^{(i)}$-th $\mathsf{BS.Sign}_2$ query, which happens before $\mathcal{A}$'s i-th special query.
  - In case (b), $\mathcal{A}$ has already seen $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(\ell+1)}$, and $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(\ell)}$ are fixed when $\mathcal{A}$ makes the ℓ-th $\mathsf{BS.Sign}_2$ query, which happens before $\mathcal{A}$'s i-th special query.

  Hence, in both cases, all $b_j^{(i)}, b_j^{(i)} c_j^{(k_j^{(i)})}$, except $\mathbf{c}^{*(i)}$, are fixed when $\mathcal{A}$ makes its i-th special query. Since $\mathbf{c}^{*(i)}$ is uniformly random, one has $\Pr[c_j^{*(i)} =$

$b_j^{(i)} c_j^{(k_j^{(i)})} \vee c_j^{*(i)} = b_j^{(i)}$ for $j \in [\lambda]] = \frac{1}{2^\lambda}$ for one query $\mathsf{H}(\mathbf{R}^{(i)} \| m_i)$. Since $\mathcal{A}$ makes at most q_h queries, we have that

$$\Pr[E \wedge E_1] \leq \frac{q_h}{2^\lambda}.$$

– Case E_2: Suppose that all special queries are made during some session which is eventually closed. Since there are ℓ such sessions and $\ell + 1$ special queries, there is at least one session, say j_0-th, with at least two special queries, say i-th and $(i+1)$-th, during it. Consider the time when $\mathcal{A}$ makes its $(i+1)$-th special query. At this point, $\mathcal{A}$ has already seen $\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(j_0)}$. Furthermore, $(I^{(i)}, \mathbf{z}^{(i)}, \mathbf{b}^{(i)})$ and $(I^{(i+1)}, \mathbf{z}^{(i+1)}, \mathbf{b}^{(i+1)})$, with $k_j^{(i)}, k_j^{(i+1)} \leq j_0$ for $j \in [\lambda]$, are all fixed. The output $\mathcal{A}$'s i-th special query $\mathbf{c}^{*(i)}$ is also fixed right after $\mathcal{A}$ makes its i-th special query, which happens before $\mathcal{A}$'s $(i+1)$-th special query. Moreover, $\mathbf{c}^{(j)}$ (where $j \in [j_0 - 1]$) is fixed when $\mathcal{A}$ makes its j-th $\mathsf{BS.Sign}_2$ query, which happens before $\mathcal{A}$'s $(i+1)$-th special query which is made during the j_0-th session starting after the j-th session for $j \in [j_0 - 1]$. That means that when $\mathcal{A}$ makes its $(i+1)$-th special query, only $\mathbf{c}^{(j_0)}$ and $\mathbf{c}^{*(i+1)}$ are unfixed.
Now, we consider the time when $\mathcal{A}$ makes its j_0-th $\mathsf{BS.Sign}_2$ query, in other words, when the j_0-th session is closed. At this point $\mathbf{c}^{*(i+1)}$ is also fixed, so the only unfixed is $\mathbf{c}^{(j_0)}$ which is chosen by $\mathcal{A}$.
If either $k_j^{(i)} \neq j_0$ or $k_j^{(i+1)} \neq j_0$ for some $j \in [\lambda]$ then the probability that $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})} \vee c_j^{*(i)} = b_j^{(i)}$ and $c_j^{*(i+1)} = b_j^{(i+1)} c_j^{(k_j^{(i+1)})} \vee c_j^{*(i+1)} = b_j^{(i+1)}$ $\forall\, j \in [\lambda]$ is at most $\frac{1}{2^\lambda}$ over the choice of $\mathbf{c}^{*(i)}$ and $\mathbf{c}^{*(i+1)}$. We now assume that $k_j^{(i)} = k_j^{(i+1)} = j_0$ for all $j \in [\lambda]$. In this case, the probability that that $c_j^{*(i)} = b_j^{(i)} c_j^{(j_0)} \vee c_j^{*(i)} = b_j^{(i)}$ and $c_j^{*(i+1)} = b_j^{(i+1)} c_j^{(j_0)} \vee c_j^{*(i+1)} = b_j^{(i+1)}$ $\forall\, j \in [\lambda]$ is at most $\frac{1}{2^\lambda}$. This is because when $\mathcal{A}$ makes its $(i+1)$-th special query, $\mathbf{c}^{*(i+1)}$ is uniformly random and others are fixed.
Since $\mathcal{A}$ makes q_h random oracle queries in total, we have that

$$\Pr[E \wedge E_2] \leq \frac{\binom{q_h}{2}}{2^\lambda}.$$

In summary, we have

$$\Pr[E] \leq \frac{1}{2^\lambda} + \frac{q_h}{2^\lambda} + \frac{\binom{q_h}{2}}{2^\lambda} = \frac{q_h^2 + 3q_h + 2}{2^{\lambda+1}}.$$

This implies that

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} \geq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} - \frac{q_h^2 + 3q_h + 2}{2^{\lambda+1}}.$$

Reduction to OM-GA-DLOG. We now design an adversary $\mathcal{B}$ to show a reduction from OM-GA-DLOG to Game_1. Let $\mathcal{B}$ run on input $\mathsf{CEGAT} = (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$ and have oracle access to chal and gadlog. $\mathcal{B}$ first queries $E_1 := \mathsf{chal}()$ and runs $\mathcal{A}$ on input $(\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, E_1)$. $\mathcal{B}$ runs the code of Game_1 except:

(1) on $\mathcal{A}$'s j-th $\mathsf{BS.Sign}_1$, for $j \in [\ell]$, query $\mathcal{B}$ returns $\mathbf{Y}^{(j)} := \mathsf{chal}()$;
(2) on $\mathcal{A}$'s j-th $\mathsf{BS.Sign}_2(\mathbf{c}^{(j)})$ query, $\mathcal{B}$ returns $\mathbf{s}^{(j)} := \mathsf{gadlog}(E_1^{\mathbf{c}^{(j)}}, \mathbf{Y}^{(j)})$. Note that for each such j-th query to $\mathsf{BS.Sign}_2$, $\mathcal{B}$ has to make λ queries to gadlog. Hence, for ℓ query of $\mathcal{A}$ to $\mathsf{BS.Sign}_2$, $\mathcal{B}$ has to make totally $\ell\lambda$ queries to gadlog oracle.

$\mathcal{B}$ answers $\mathcal{A}$'s $(\ell+1)$-th $\mathsf{BS.Sign}_1$ query just as in Game_1, i.e., by sampling $\mathbf{y}_{\ell+1} \in \mathbb{Z}_N^\lambda$ and returning $\mathbf{Y}^{(\ell+1)} := \mathbf{y}_{\ell+1} \diamond E_0$. Finally, when $\mathcal{A}$ returns its final outputs $(m_1, (\mathbf{R}^{(1)}, \mathbf{r}^{(1)})), \ldots, (m_{\ell+1}, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)}))$, if there exist some $i \in [\ell+1]$ and $j \in [\lambda]$ such that $c_j^{*(i)} \neq b_j^{(i)} c_j^{(k_j^{(i)})}$ or $c_j^{*(i)} \neq b_j^{(i)}$, then $\mathcal{B}$ computes x as either

$$x := \frac{z_j^{(i)} + b_j^{(i)} r_{I_j^{(i)}}^{(k_j^{(i)})} - r_j^{(i)}}{c_j^{*(i)} - b_j^{(i)} c_j^{(k_j^{(i)})}} \quad \text{or} \quad x := \frac{z_j^{(i)} - r_j^{(i)}}{c_j^{*(i)} - b_j^{(i)}} \tag{19}$$

and

$$\mathbf{r}^{*(i)} := \mathbf{s}^{(i)} + \mathbf{c}^{(i)} \cdot x$$

for $i \in [\ell]$. It is clear that $x = \mathsf{gadlog}(E_0, E_1)$ and $\mathbf{r}^{*(i)} = \mathsf{gadlog}(E_1, \mathbf{Y}^{(i)})$ for $i \in [\ell]$. Altogether, $\mathcal{B}$ has obtained totally $\ell\lambda + 1$ solutions for $\mathsf{GA\text{-}DLOG}$. Hence $(x, \mathbf{r}^{*(1)}, \ldots, \mathbf{r}^{*(\ell)})$ is the solution for $\mathsf{OM\text{-}GA\text{-}DLOG}$ problem, i.e., $\mathcal{B}$ wins the $\mathsf{OM\text{-}GA\text{-}DLOG}$. Therefore, we have that

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{OM\text{-}GA\text{-}DLOG}}(\lambda) = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda).$$

We conclude that

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{OM\text{-}GA\text{-}DLOG}}(\lambda) \geq \mathbf{Adv}_{\mathsf{BS},\ell,1,\mathcal{A}}^{\mathsf{OMUF}}(\lambda) - \frac{q_h^2 + 3q_h + 2}{2^{\lambda+1}}$$

which completes the proof. □

Concrete-security note. Theorem 3 is an asymptotic security statement against polynomial-time adversaries. For concrete parameters, one should also account for the sequential mix-and-match attack [26], which uses one signing session and about $2^{\lambda/2}$ hash evaluations when the base challenge space has size two. Thus, for n-bit classical concrete security in the sequential setting, the repetition parameter should be chosen with margin, e.g., $\lambda \geq 2n$. Larger parameters may be required when quantum search is considered.

Remark on two-open-session concurrency. We do not claim $(\ell, 2)$-OMUF security for polynomially many total signing sessions. The reason is that the scheme obtains the challenge space $\{-1, 1\}^\lambda$ by parallel repetition of binary-challenge components. The mix-and-match attack of Do, Hanzlik, and Paracucchi [26, Section 3.3] applies to such repeated Schnorr-type blind signatures: with at most two sessions open at any time, but with λ total signing sessions, the adversary can combine simulated transcript coordinates and real signer responses to produce $\lambda + 1$ valid signatures. Thus our positive concurrent security result is restricted to the logarithmic-session regime.

5.3 Logarithmic-Session Concurrent One-More Unforgeability

In this section, we provide a proof for logarithmic-session concurrent security of the Schnorr blind signature BS in Figure 6 in AGAM + ROM, following the framework of the proof in [37] with adaptation to our setting. Here logarithmic-session means that the total number of signing sessions is logarithmic in the security parameter. First of all, we define the *parallel binary Random Inhomogeneities in an Overdetermined Solvable system of linear equations* (termed pROS) problem, as akin to that in [37,43], in our setting.

Definition 17 (pROS Problem). *Given the parameters $\ell, \lambda \in \mathbb{N}$, let $H_{pROS} : \{-1,0,1\}^{\lambda\times(\ell\lambda+1)} \times \{0,1\}^* \to \{-1,1\}^\lambda$ be a random oracle. The pROS problem is given access to H_{pROS}, and the adversary $\mathcal{A}$ outputs:*

- *$\ell+1$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\ell+1} \in \{-1,0,1\}^{\lambda\times(\ell\lambda+1)}$,*
- *a secret challenge vector $\mathbf{c} = (1\ \mathbf{c}')^\top$ where $\mathbf{c}' \in \{-1,1\}^{\ell\lambda}$,*
- *$\ell+1$ strings $\mathsf{aux}_1, \ldots, \mathsf{aux}_{\ell+1} \in \{0,1\}^*$.*

We say $\mathcal{A}$ wins if $\mathcal{A}$'s output satisfies the following conditions:

1. *Each row of $\mathbf{A}_1, \ldots, \mathbf{A}_{\ell+1}$ has one and only one non-zero entry from $\{-1,1\}$;*
2. $$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_\ell \\ \mathbf{A}_{\ell+1} \end{bmatrix} \cdot \mathbf{c} = \begin{bmatrix} H_{pROS}(\mathbf{A}_1, \mathsf{aux}_1) \\ \vdots \\ H_{pROS}(\mathbf{A}_\ell, \mathsf{aux}_\ell) \\ H_{pROS}(\mathbf{A}_{\ell+1}, \mathsf{aux}_{\ell+1}) \end{bmatrix},$$

We denote by $\mathbf{Adv}^{pROS}_{\ell,\mathcal{A}}(\lambda)$ the advantage of an adversary $\mathcal{A}$ in winning this problem.

Note that λ refers to the number of parallel repetitions in our scheme and ℓ refers to the total number of signing sessions considered in the logarithmic-session concurrent security experiment. Now we show an upper bound on the advantage that adversaries can achieve in the pROS problem, where the proof is also inspired by [37].

Theorem 4. *For any computationally-unbounded adversary $\mathcal{A}$ of the pROS problem with Q random oracle queries, we have*

$$\mathbf{Adv}^{pROS}_{\ell,\mathcal{A}}(\lambda) \leq \frac{\binom{Q}{\ell+1}}{2^\lambda}$$

Proof. First, consider the case where $\mathcal{A}$ makes fewer than $\ell+1$ queries, i.e., $Q < \ell+1$. To win, $\mathcal{A}$ must output $\ell+1$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\ell+1}$ and corresponding auxiliary strings $\mathsf{aux}_1, \ldots, \mathsf{aux}_{\ell+1}$. If $Q < \ell+1$, at least one of the pairs $(\mathbf{A}_k, \mathsf{aux}_k)$ output by $\mathcal{A}$ must not have been queried to $\mathsf{H}_{\mathsf{pROS}}$. Let $(\mathbf{A}_k, \mathsf{aux}_k)$ be such a pair. The output $h_k = \mathsf{H}_{\mathsf{pROS}}(\mathbf{A}_k, \mathsf{aux}_k)$ is then a uniformly random and independent vector in $\{-1,1\}^\lambda$ from $\mathcal{A}$'s perspective. The winning condition requires the

existence of a secret vector $\mathbf{c}' \in \{-1,1\}^{\ell\lambda}$ such that, among other conditions, $\mathbf{A}_k \cdot \mathbf{c} = h_k$, where $\mathbf{c} = (1\ \mathbf{c}')^\top$. The probability that a random h_k matches the value required by the chosen $\mathbf{c}'$ and $\mathbf{A}_k$ is at most $1/2^\lambda$. More importantly, finding a single $\mathbf{c}'$ satisfying all $\ell+1$ equations when one output h_k is uniformly random is highly improbable. The probability in this case is certainly bounded by the expression derived below. Thus, we assume $Q \geq \ell + 1$.

Let $\mathcal{A}$ make $Q \geq \ell + 1$ queries. To win, $\mathcal{A}$ must choose $\ell + 1$ distinct pairs $(\mathbf{A}_1, \mathsf{aux}_1), \ldots, (\mathbf{A}_{\ell+1}, \mathsf{aux}_{\ell+1})$ from its queries and find a secret vector $\mathbf{c}' \in \{-1,1\}^{\ell\lambda}$ such that $\mathbf{A}_i \cdot \mathbf{c} = \mathsf{H}_{\mathsf{pROS}}(\mathbf{A}_i, \mathsf{aux}_i)$ for all $i \in \{1, \ldots, \ell+1\}$. Consider an arbitrary fixed set of $\ell + 1$ distinct queries $\{(\mathbf{A}_i, \mathsf{aux}_i)\}_{i=1}^{\ell+1}$ made by $\mathcal{A}$. Let $h_i = \mathsf{H}_{\mathsf{pROS}}(\mathbf{A}_i, \mathsf{aux}_i)$ be the corresponding outputs. Since $\mathsf{H}_{\mathsf{pROS}}$ is a random oracle and the queries are distinct, the output tuple $\mathbf{h} = (h_1, \ldots, h_{\ell+1})$ is uniformly distributed in the target space $T = (\{-1,1\}^\lambda)^{\ell+1}$. The size of this space is $|T| = (2^\lambda)^{\ell+1}$. The adversary wins with this specific set of queries if there exists a secret vector $\mathbf{c}'$ in the solution space $S = \{-1,1\}^{\ell\lambda}$ such that the equation $\mathbf{M} \cdot \mathbf{c} = \mathbf{h}$ holds, where $\mathbf{M}$ is the $((\ell+1)\lambda) \times (\ell\lambda + 1)$ matrix formed by stacking $\mathbf{A}_1, \ldots, \mathbf{A}_{\ell+1}$, $\mathbf{c} = (1\ \mathbf{c}')^\top$, and $\mathbf{h}$ is the vector formed by stacking $h_1, \ldots, h_{\ell+1}$. The size of the solution space is $|S| = 2^{\ell\lambda}$. Let $f : S \to T$ be the map defined by $f(\mathbf{c}) = \mathbf{M} \cdot \mathbf{c}$. The image of this map, $\mathrm{Im}(f)$, contains all possible outcomes that can be generated by some secret vector $\mathbf{c}'$. The size of the image is bounded by the size of the domain $|\mathrm{Im}(f)| \leq |S| = 2^{\ell\lambda}$. The probability that the uniformly random output $\mathbf{h}$ from the random oracle falls into this image space (for this fixed set of queries) is

$$\Pr[\mathbf{h} \in \mathrm{Im}(f)] = \frac{|\mathrm{Im}(f)|}{|T|} \leq \frac{2^{\ell\lambda}}{(2^\lambda)^{\ell+1}} = \frac{1}{2^\lambda}.$$

The adversary $\mathcal{A}$ can choose any subset of $\ell + 1$ distinct queries out of its Q total queries. The number of ways to choose such a subset is $\binom{Q}{\ell+1}$. By the union bound, the total probability that $\mathcal{A}$ wins is bounded by

$$\mathbf{Adv}^{\mathsf{pROS}}_{\ell,\mathcal{A}}(\lambda) \leq \frac{\binom{Q}{\ell+1}}{2^\lambda},$$

which concludes the proof. □

Theorem 5. *Given a PPT adversary $\mathcal{A}$ against the Schnorr blind signature scheme BS in Figure 6 over the random oracle model and the algebraic group action model, there exist a pROS adversary $\mathcal{B}_1$ and an OM-GA-DLOG adversary $\mathcal{B}_2$ such that*

$$\mathbf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\ell,\eta,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{pROS}}_{\ell,\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathsf{OM\text{-}GA\text{-}DLOG}}_{\mathcal{B}_2}(\lambda).$$

It follows from Theorem 4 that if we assume the number of oracle queries $Q = \mathrm{poly}(\lambda)$, then setting $\ell = O(\log(\lambda))$ will make $\binom{Q}{\ell+1}\frac{1}{2^\lambda}$ negligible. Consequently, the scheme satisfies logarithmic-session concurrent one-more unforgeability in AGAM + ROM under the OM-GA-DLOG assumption.

The proof follows the framework as in that of [37, Theorem 4] and [34, Theorem 2]. Again, the proof shares the key idea in that of our Theorem 3 until estimating the probability that $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})}$ or $c_j^{*(i)} = b_j^{(i)}$ for $i \in [\ell+1]$ and $j \in [\lambda]$. In these special cases, we can reduce it to the pROS problem bounded by Theorem 4. The details are as follows.

Proof. We proceed the proof in a sequence of games Game_0-Game_1 in which Game_0 is the $\mathsf{OMUF}^{\mathcal{A}}_{\mathsf{BS},\ell,\eta}(\lambda)$ game in Figure 2. We let $\mathbf{Adv}^{\mathsf{Game}_i}_{\mathcal{A}}(\lambda)$ be the advantage of $\mathcal{A}$ winning the Game_i for $i \in \{0,1\}$. Thus,

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\ell,\eta,\mathcal{A}}(\lambda).$$

Now we introduce the game hop in detail.

Game_0. This is the original $\mathsf{OMUF}^{\mathcal{A}}_{\mathsf{BS},l,\eta}(\lambda)$ game where the algebraic adversary $\mathcal{A}$ makes ℓ signing queries. During the game, $\mathcal{A}$ obtains $1+\ell\lambda$ set elements $(E_1, Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_\lambda^{(\ell)})$, where for each $i \in [\ell]$, $\mathbf{Y}^{(i)} = (Y_j^{(i)} : j \in [\lambda])$ is obtained from the i-th signing session returned by $\mathsf{BS.Sign}_1$. Let $(m_1, (\mathbf{R}^{(1)}, \mathbf{r}^{(1)}))$, $\cdots$, $(m_{\ell+1}, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)}))$ be the output of $\mathcal{A}$, i.e., $(\mathbf{R}^{(i)}, \mathbf{r}^{(i)})$ is the forgery of m_i for $i \in [\ell+1]$. Additionally, since $\mathcal{A}$ is algebraic, for each $i \in [\ell+1]$, it needs to output also $(I^{(i)}, \mathbf{z}^{(i)}, \mathbf{b}^{(i)})$ with $I^{(i)} = \{(k_j^{(i)}, I_j^{(i)}) \in [\ell+1] \times [\lambda] : j \in [\lambda]\}$, $\mathbf{z}^{(i)} \in \mathbb{Z}_N^\lambda$, and $\mathbf{b}^{(i)} \in \{1,-1\}^\lambda$.

Game_1. The only difference between Game_1 and Game_0 lies in the simulation of the random oracle via the standard lazy-sampling method and an additional abort condition appended to the end of the game. When the adversary makes a query $\mathsf{H}(\mathbf{R}\|m)$, we first record its representation $(\mathbf{I}, \mathbf{z}, \mathbf{b})$. If the query is new, we respond with a string sampled uniformly at random from the output space.

Upon receiving the forgeries $(m_1, (\mathbf{R}^{(1)}, \mathbf{r}^{(1)}))$, $\cdots$, $(m_{\ell+1}, (\mathbf{R}^{(\ell+1)}, \mathbf{r}^{(\ell+1)}))$ from $\mathcal{A}$, we abort the game if for each $i \in [\ell+1]$ and $j \in [\lambda]$, either $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})}$ or $c_j^{*(i)} = b_j^{(i)}$. Define the matrix $\mathbf{A}_i \in \{-1,0,1\}^{\lambda \times (\ell\lambda+1)}$ for each $i \in [\ell+1]$ as follows: for each row $j \in [\lambda]$, set either the 1-th entry to $b_j^{(i)}$ (to capture $c_j^{*(i)} = b_j^{(i)}$) or the $(\lambda(k_j^{(i)}-1)+j+1)$-th entry to $b_j^{(i)}$ (to capture $c_j^{*(i)} = b_j^{(i)} c_j^{(k_j^{(i)})}$), and set all others to zero. Then the aborting condition is equivalent to satisfy

$$\mathbf{A}_i \cdot \left(1\ c_1^{(1)} \cdots\ c_\lambda^{(1)}\ \cdots\ c_1^{(\ell)} \cdots c_\lambda^{(\ell)}\right)^\top = \mathsf{H}(\mathbf{R}^{(i)} || m_i) \tag{20}$$

for each $i \in [\ell+1]$, Note that simulating the random oracle does not alter the distribution, and such a modification in the game does not affect the adversary's advantage, except in the case where the adversary can solve the pROS problem. Then we can construct the pROS problem adversary $\mathcal{B}_1$ in this way: $\mathcal{B}_1$ simulates the (ℓ,η)-OMUF by itself, simulates $\mathsf{H}_{\mathsf{pROS}}$ by using H described in the

construction, and interacts with $\mathcal{A}$. Once $\mathcal{A}$ returns the forgeries, $\mathcal{B}_1$ establishes the relations as in Eq. (20). By the Difference Lemma [55], this implies that

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \geq \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) - \mathbf{Adv}^{\mathsf{pROS}}_{\ell,\mathcal{B}_1}(\lambda).$$

Now given an adversary $\mathcal{A}$ in Game_1, we can construct an algorithm $\mathcal{B}_2$ solving the OM-GA-DLOG as in that of Theorem 3, which yields

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{OM\text{-}GA\text{-}DLOG}}_{\mathcal{B}_2}(\lambda).$$

Therefore, we can conclude that

$$\mathbf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\ell,\eta,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{pROS}}_{\ell,\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathsf{OM\text{-}GA\text{-}DLOG}}_{\mathcal{B}_2}(\lambda).$$

□

6 Schnorr-Signed Hashed ElGamal KEM

In this section, we provide a construction of a Schnorr-signed Hashed ElGamal KEM from CSIDH group action, which is depicted in Figure 8. We will prove that the KEM is tightly secure under the hardness of the GA-DLOG problem in AGAM + ROM. The description of the scheme consists of the following algorithms.

- $\mathsf{KEM.Setup}(1^\lambda)$: On input the security parameter λ, it invokes the group generator GrGen to generate a $\mathsf{CEGA} = (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0)$, selects two hash functions $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}' : \{0,1\}^* \to \mathcal{K}$, where $\mathcal{K}$ is the key space, and outputs $(\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, \mathsf{H}, \mathsf{H}')$ as a public parameter par.
- $\mathsf{KEM.KeyGen}(\mathsf{par})$: On input the public parameter $\mathsf{par} := (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, \mathsf{H}, \mathsf{H}')$, samples $y \in \mathbb{Z}_N$ and computes $Y := y \diamond E_0$. It sets $\mathsf{sk} := (\mathsf{par}, y)$, $\mathsf{pk} := (\mathsf{par}, Y)$, and returns $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{KEM.Enc}(\mathsf{pk})$: On input the public key $\mathsf{pk} = (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, \mathsf{H}, \mathsf{H}', Y)$, samples $x \xleftarrow{\$} \mathbb{Z}_N, \mathbf{r} \xleftarrow{\$} Z_N^\lambda$ and computes $E_1 := x \diamond E_0$ and $\mathbf{R} := \mathbf{r} \diamond E_0$. Next it computes the key $K := \mathsf{H}'(x \diamond Y)$, $\mathbf{c} := \mathsf{H}(\mathbf{R}, E_1)$, and a Schnorr signature $\mathbf{s} := \mathbf{r} - \mathbf{c} \cdot x$. It outputs $(K, (E_1, \mathbf{R}, \mathbf{s}))$.
- $\mathsf{KEM.Dec}(\mathsf{sk}, (E_1, \mathbf{R}, \mathbf{s}))$: On input the secret key $\mathsf{sk} := (\mathbb{Z}_N, \mathcal{X}, \diamond, E_0, \mathsf{H}, \mathsf{H}', y)$, and $(E_1, \mathbf{R}, \mathbf{s})$, computes $\mathbf{c} := \mathsf{H}(\mathbf{R}, E_1)$. It then set $\mathbf{R}' \in \mathcal{X}^\lambda$ with $R'_i := s_i \diamond E_{c_i}$ for each $i \in [\lambda]$. If $\mathbf{R}' \neq \mathbf{R}$ then it returns $\perp$; otherwise it returns the key computed by $K := \mathsf{H}'(y \diamond X)$.

The correctness of the Schnorr-signed hashed ElGamal KEM in Figure 8 is straightforward. We now prove that it is tightly IND-CCA2-secure in the AGAM + ROM under the hardness of the GA-DLOG assumption.

Theorem 6. *Let GrGen be a group generator. Let $\mathcal{A}$ be an algebraic adversary against the IND-CCA2 security of the Schnorr-signed ElGamal KEM scheme in Figure 8, making at most q_d decryption queries and q_h queries to both random oracles. Then, there exists an algorithm $\mathcal{B}$ solving the GA-DLOG problem with respect to GrGen, such that*

$$\mathbf{Adv}^{\mathit{IND\text{-}CCA2}}_{\mathit{KEM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathit{GA\text{-}DLOG}}_{\mathcal{B}}(\lambda) + \frac{q_h + 7q_d}{2^{\lambda+1}}.$$

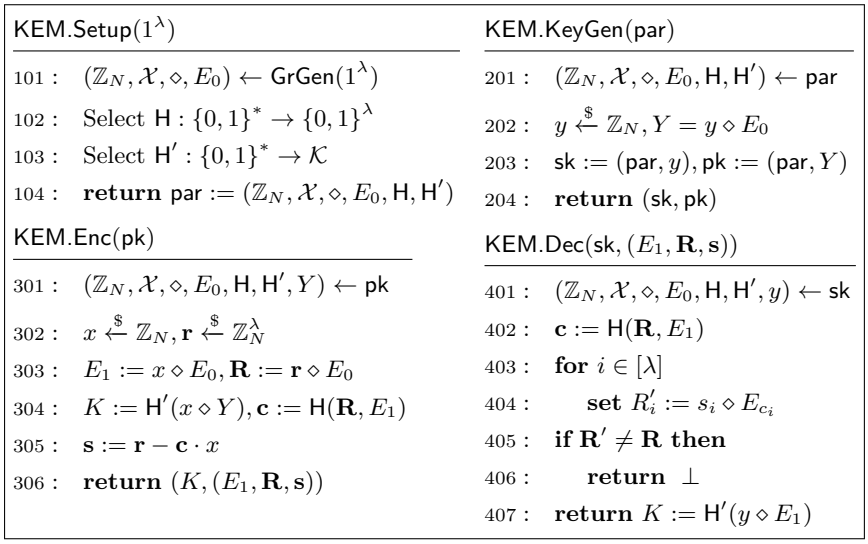

Fig. 8. The Schnorr-Signed Hashed ElGamal KEM from CEGA

The proof follows the framework of [34, Theorem 6] with adaptation to the group action context. Let us sketch the idea of the proof first, which follows the framework of [34, Theorem 6]. Given a GA-DLOG challenge Y, we set it as the public key $\mathsf{pk} := Y$, pick a random $z \xleftarrow{\$} \mathbb{Z}_N$ and set $E_1^* := z \diamond Y$. Let $(E_1^*, \mathbf{R}^*, \mathbf{s}^*)$ be a challenge ciphertext. If the adversary makes the query $\mathsf{H}'(K^*)$ then we have $K^* = (z + 2y) \diamond E_0$. Since $\mathcal{A}$ is algebraic, it must provide a representation $(u, j) \in \mathbb{Z}_N \times [\lambda]$ such that $(z+2y) \diamond E_0$ is equal to either (1) $u \diamond E_0$, or (2) $u \diamond Y$, or (3) $u \diamond E_1^*$, or (4) $u \diamond R_j^*$. We show that in each case, we can easily obtain the GA-DLOG solution y for Y, i.e., $Y = y \diamond E_0$.

(1) In the case that $(z + 2y) \diamond E_0 = u \diamond E_0$, we can easily obtain $y = \frac{u-z}{2}$ given that 2 is co-prime with N, which is always true for CSIDH-512 parameters.
(2) In the case that $(z+2y) \diamond E_0 = u \diamond Y = (u+y) \diamond E_0$, we can obtain $y = u - z$.
(3) In the case that $(z + 2y) \diamond E_0 = u \diamond E_1^* = (u + z + y) \diamond E_0$, we can obtain $y = u$.
(4) Now consider the case that $(z+2y) \diamond E_0 = u \diamond R_j^*$. Note that $R_j^* = s_j^* \diamond E_{c_j^*}^* = (s_j^* + c_j^* \cdot (z + y)) \diamond E_0$. We have that

$$\begin{aligned}(z + 2y) \diamond E_0 &= u \diamond R_j^* \\ &= u \diamond ((s_j^* + c_j^* \cdot (z + y)) \diamond E_0) \\ &= (u + s_j^* + c_j^* \cdot (z + y)) \diamond E_0\end{aligned}$$

which implies that we can obtain y by

$$y = \frac{u + s_j^* + (c_j^* - 1)z}{2 - c_j^*}$$

as long as $2 - c_j^*$, with $c_j^* \in \{0, 1\}$, is invertible modulo N, which is again always true for CSIDH-512 parameters.

The rest of the game is to simulate the game without the knowledge of GA-DLOG y of Y, and thereby also without the knowledge of GA-DLOG of $\mathbf{X}^*$, which we will proceed as follows:

- The Schnorr signature $\mathbf{s}^*$ under E_1^* contained in the challenge encapsulation can be simulated by programming the random oracle H as in the proof of Theorem 1.
- For the decryption queries, say on $(E_1, \mathbf{R}, \mathbf{s})$, we need to extract the knowledge of x such that $E_1 = x \diamond E_0$, so that we can answer the query by returning $K = \mathsf{H}'(x \diamond Y)$. However, the extraction is not as simple as in the proof of Theorem 1, since both E_1 and $\mathbf{R}$ can also depend on Y, E_1^* and $\mathbf{R}^*$ in case the query is made after seeing the challenge ciphertext. So we need to consider the representation of E_1 in terms of E_0, Y, E_1^* and $\mathbf{R}^*$. To be more specific, we will do the following.

(1) If $E_1 = u \diamond E_0$ then we can obtain $x := u$.
(2) If $E_1 = u \diamond Y = (u+y) \diamond E_0$. Since $(E_1, \mathbf{R}, \mathbf{s})$ is a valid query, we have, for each $i \in [\lambda]$, $R_i = s_i \diamond E_{c_i}$, where $\mathbf{c} = \mathsf{H}(\mathbf{R}, E_1)$. Hence $R_i = (s_i + c_i u + c_i y) \diamond E_0$. We need to consider the representation of R_i in terms of E_0, Y, E_1^* and $\mathbf{R}^*$.
   (i) If $R_i = v_i \diamond E_0$ then we can solve y if $c_i = 1$ and abort otherwise.
   (ii) If $R_i = v_i \diamond Y = (v_i + y) \diamond E_0$, then we can find y if $c_i = 0$ and abort otherwise.
   (iii) If $R_i = v_i \diamond E_1^* = (v_i + z + y) \diamond E_0$, then again we can find y if $c_i = 0$ and abort otherwise.
   (iv) If $R_i = v_i \diamond R_j^* = (v_i + s_j^* + c_j^* \cdot (z + y)) \diamond E_0$, then we can find y of $c_i \neq c_j^*$ and abort otherwise.

   We see that if for all $i \in [\lambda]$, we fall in the aborting situation in either case from (i) to (iv), then we are unable to find y; otherwise, we can find y in at least one of non-aborting case. Thus, the probability for aborting is $\frac{1}{2^\lambda}$.
(3) If $E_1 = u \diamond E_1^* = (u + z + y) \diamond E_0$, then we can do similarly to case (2).
(4) If $E_1 = u \diamond R_j^* = (u + s_j^* + c_j^* \cdot (z + y)) \diamond E_0$, then consider if $c_j^* = 0$, in which we can obtain $x := u + s_j^*$. If $c_j^* = 1$, then we can again argue as similar to case (2).

In conclusion, we can always either extract x such that $E_1 = x \diamond E_0$ which can help us answer the decryption query, or solve the GA-DLOG solution y for the public key Y, in which case the reduction stops, except in the aborting cases that happen with probability bounded by $\frac{3}{2^\lambda}$.

For convenience in the formal proof later, we define the following aborting condition when answering the decryption query with $(E_1, \mathbf{R}, \mathbf{s}$ and (u, j) the representation of E_1 in terms of E_0, Y, E_1^* and $\mathbf{R}^*$.

(I) $E_1 \neq u \diamond E_0$.
(II) All aborting cases that happen in (2)–(4) above.

(III) Not all aborting cases in (2)–(4) above.

Proof of Theorem 6. The proof follows the same framework of [34, Theorem 6], which is done via a series of games through Game_0 to Game_3. We define $\mathbf{Adv}^{\mathsf{Game}_i}_{\mathcal{A}}(\lambda)$ as the advantage of $\mathcal{A}$ in winning Game_i, for each $i \in [3]$.

Game_0. This game, depicted in Figure 9, has the same behavior as the IND-CCA2 game in Figure 3; the only change is that the value E_1^* used in Enc oracle is already set before running $\mathcal{A}$ to ensure that in later games it is defined in the abort conditions. It is clear that

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM},\mathcal{A}}(\lambda).$$

Game_0: Generate $(\mathbb{Z}_N, \mathcal{X}, \diamond, E_0) \leftarrow \mathsf{GrGen}(1^\lambda)$, sample $y \xleftarrow{\$} \mathbb{Z}_N$, set $Y := y \diamond E_0$, and sample $b \xleftarrow{\$} \{0,1\}$.
T, T' are lazy-sampled tables for $\mathsf{H}(\mathbf{R}, E_1)$ and $\mathsf{H}'(K)$.
Enc samples $x^* \xleftarrow{\$} \mathbb{Z}_N$, sets $E_1^* := x^* \diamond E_0$, computes $(\mathbf{R}^*, \mathbf{s}^*)$, and returns $(K_b, (E_1^*, \mathbf{R}^*, \mathbf{s}^*))$.
Dec rejects $(E_1^*, \mathbf{R}^*, \mathbf{s}^*)$, verifies $\mathbf{s} \diamond E_{\mathbf{c}} = \mathbf{R}$ for $\mathbf{c} := \mathsf{H}(\mathbf{R}, E_1)$, and returns $\tilde{\mathsf{H}}(y \diamond E_1)$.

Fig. 9. Description of Game_0 in the proof of Theorem 6.

Game_1. This game is different from Game_0 in that, it first sets $\mathsf{U} := ()$ at the beginning before calling the adversary $\mathcal{A}$ to be the set recording the representation $[u, j, u', j']$ of E_1 and $\mathbf{R}$ in terms of $E_0, Y, E_1^*, \mathbf{R}^*$ whenever the adversary makes some queries $\mathsf{H}(E_1, \mathbf{R})$. It also sets $\mathbf{c}^* := 0$ during that phase such that the value is defined in Dec when Enc has not been called yet.

As a similar argument in [34], at any point $\mathsf{T}(\mathbf{R}^*, E_1^*)$ is the only value in T that might not have been set during adversary's query to H of Dec, and that could not have a corresponding entry in U. In addition, if $(E_1, \mathbf{R}, \mathbf{s})$ is queried to Dec which has answer not a $\perp$, then we must have $(E_1, \mathbf{R}) \neq (E_1^*, \mathbf{R}^*)$; since otherwise, according to line 703 in Figure 9, we would have $\mathbf{s} \diamond E_{\mathbf{c}} = \mathbf{R} = \mathbf{R}^* = \mathbf{s}^* \diamond E_{\mathbf{c}}$ which implies that $\mathbf{s} = \mathbf{s}^*$, we will have aborting due to line 701.

The main difference between Game_0 and Game_1 is that during the decryption query, Game_1 aborts when the aborting condition (II) happens as mentioned above in the proof sketch, and returns a random bit $b' \xleftarrow{\$} \{0,1\}$. Note that Game_1 records the value $[u, j, u', j']$, the representation of E_1 and $\mathbf{R}$, when $\mathsf{T}(E_1, \mathbf{R}) = \mathbf{c}$ was randomly drawn. This means that the value $\mathbf{c}$ is independent of $[u, j, u', j']$. Hence, the above aborting event happens with probability $\frac{3}{2^\lambda}$. There are at most q_d queries to Dec, the probability of aborting is thus bounded by $\frac{3q_d}{2^\lambda}$. Therefore, we have

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \geq \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) - \frac{3q_d}{2^\lambda}.$$

Game_2. This game is similar to Game_1, except that it uses the simulation for the Schnorr signature to simulate the Enc oracle as shown below.

(1) Sample $\mathbf{c}^* \xleftarrow{\$} \{-1,1\}^\lambda, \mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_N^\lambda$ and compute $\mathbf{R}^* := \mathbf{s} \diamond E_{\mathbf{c}^*}$.
(2) If $\mathsf{T}(\mathbf{R}^*, E_1^*) = \perp$ then $\mathsf{T}(\mathbf{R}^*, E_1^*) = \mathbf{c}^*$; otherwise abort the game and return $b' \xleftarrow{\$} \{0,1\}$.
(3) It sets $K_0 = \tilde{\mathsf{H}}(x^* \diamond Y), K_1 \xleftarrow{\$} \mathcal{K}$.
(4) It finally returns $(K_b, (E_1^*, \mathbf{R}^*, \mathbf{s}^*))$.

We observe that Game_1 and Game_2 are identical unless the oracle Enc generates value $(\mathbf{R}^*, E_1^*)$ that have already been assigned a value in the table T. Since $\mathcal{A}$ makes at most q_h queries to H and q_d queries to Dec, at most $q_h + q_d$ queries to the table T. Thus the probability that $(\mathbf{R}^*, E_1^*)$ collides with one of the entries is bounded by $\frac{q_h+q_d}{2^{\lambda+1}}$. Therefore we have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}(\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda) - \frac{q_h + q_d}{2^{\lambda+1}}.$$

Game_3. This game is similar to Game_2 except:

(i) In simulation of Enc above, it returns $(K_1, (E_1^*, \mathbf{R}^*, \mathbf{s}^*))$ with $K_1 \xleftarrow{\$} \mathcal{K}$.
(ii) When adversary makes query K to H', it aborts and returns $b' \xleftarrow{\$} \{0,1\}$ if $K = y \diamond E_1^*$.
(iii) When adversary makes query $(E_1, \mathbf{R}, \mathbf{s})$ to Dec oracle, if the aborting condition (III) defined above happens, it aborts and outputs a random bit $b' \xleftarrow{\$} \{0,1\}$.

It is easy to observe that in Game_3, the output of $\mathcal{A}$ is independent of b, and if it aborts, then it outputs a random bit. We then have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3}(\lambda) = 0.$$

We now show that the difference between Game_2 and Game_3 allows us to construct an algorithm $\mathcal{B}$ to solve the GA-DLOG problem.

Given a GA-DLOG challenge Y, $\mathcal{B}$ sets Y as the public key, chooses a random $z \xleftarrow{\$} \mathbb{Z}_N$ and sets $E_1^* := z \diamond Y$. It simulates $\mathsf{Enc}()$ as in Game_3. $\mathcal{B}$ simulates other oracles in Game_3 for $\mathcal{A}$ without knowledge of y and x^* as follows.

- Queries to H': when $\mathcal{A}$ makes a query $K_{[u,j,u',j']}$ where (u,j) and $(u',' j)$ is the representation of K in terms of $E_0, Y, E_1^*, \mathbf{R}^*$ respectively, $\mathcal{B}$ will check whether $K = y \diamond E_1^*$, i.e., the aborting condition (ii), by solving y as mentioned in the sketch. If it is the case, then $\mathcal{B}$ aborts and returns y. If $K \neq y \diamond E_1^*$ then K_1 still perfectly simulates $\mathsf{H}'(y \diamond E_1^*) = K_b$.
- Queries to Dec: when $\mathcal{A}$ makes a query $(E_1, \mathbf{R}, \mathbf{s})$ to the oracle $\mathsf{Dec}()$, it will return $\perp$ if $\mathbf{s} \diamond E_{\mathbf{c}} \neq \mathbf{R}$ where $\mathbf{c} = \mathsf{H}(\mathbf{R}, E_1)$, or if $(E_1, \mathbf{R}, \mathbf{s}) = (E_1^*, \mathbf{R}^*, \mathbf{s}^*)$. Since $\mathbf{s}$ is determined by $(\mathbf{R}, E_1)$, the latter implies that $(\mathbf{R}, E_1) \neq (\mathbf{R}^*, E_1^*)$. Since the oracle did not return $\perp$ in line 703 in Figure 9, we consider the aborting events happening in Game_3.
  - If (III) happens, then as mentioned above in the proof sketch, $\mathcal{B}$ can obtain a GA-DLOG solution y.

- If (II) happens, then both Game_2 and Game_3 abort.
- Otherwise (I) happens, then it can recover x and return $\mathsf{H}'(x \diamond Y)$.

This shows that whenever Game_3 differs from Game_2, reduction $\mathcal{B}$ solves the GA-DLOG problem, which implies that

$$\mathbf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}(\lambda) \geq \mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) - \mathbf{Adv}^{\mathsf{GA\text{-}DLOG}}_{\mathcal{B}}(\lambda).$$

In summary, we conclude that

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{GA\text{-}DLOG}}_{\mathcal{B}}(\lambda) + \frac{q_h + 7q_d}{2^{\lambda+1}}$$

which completes the proof. □

References

1. Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 699–728. Springer, 2022.
2. Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2000.
3. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020.
4. Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani Saarinen, Paolo Santini, and Robert Wallace. Less: Linear equivalence signature scheme, 2023.
5. Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 82–99. Springer, 2013.
6. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2009.

7. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptol.*, 16(3):185–215, 2003.
8. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53. Springer, 2021.
9. Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *Algorithmic Number Theory Symposium*, 2020.
10. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492. Springer, 2020.
11. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Advances in Cryptology - ASIACRYPT 2019*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019.
12. Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. LESS is more: Code-based signatures without syndromes. In Abderrahmane Nitaj and Amr M. Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, volume 12174 of *Lecture Notes in Computer Science*, pages 45–65. Springer, 2020.
13. Markus Bläser, Dung Hoang Duong, Anand Kumar Narayanan, Thomas Plantard, Youming Qiao, Arnaud Sipasseuth, and Gang Tang. Alteq signature. https://pqcalteq.github.io/, 2023. Accessed May 2025.
14. Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. Hawk: a signature scheme inspired by the lattice isomorphism problem. https://hawk-sign.info/, 2023. Accessed May 2025.
15. Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993.
16. Gilles Brassard and Moti Yung. One-way group actions. In *Advances in Cryptology - CRYPTO 1990*, pages 94–107, 1990.
17. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.

18. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 395–427. Springer, 2018.
19. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982.
20. David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 177–182. Springer, 1988.
21. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.
22. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.
23. Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Matrix equivalence digital signature scheme. https://www.meds-pqc.org/, 2023. Accessed May 2025.
24. Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Take your meds: Digital signatures from matrix code equivalence. In *Progress in Cryptology - AFRICACRYPT 2023*, 2023.
25. Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, 2006.
26. Khue Do, Lucjan Hanzlik, and Eugenio Paracucchi. M&M'S: Mix and Match Attacks on Schnorr-Type Blind Signatures with Repetition. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 363–387. Springer, 2024.
27. Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 65–94. Springer, 2022.
28. Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryp-*

tography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part I, volume 13940 of *Lecture Notes in Computer Science*, pages 406–435. Springer, 2023.
29. Dung Hoang Duong, Thanh Xuan Khuc, Youming Qiao, Willy Susilo, and Chuanqi Zhang. Blind signatures from cryptographic group actions. *IACR Cryptol. ePrint Arch.*, page 397, 2025. To appear in ACNS 2026.
30. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 187–212. Springer, 2020.
31. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO 1986*, pages 186–194, 1986.
32. Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2010.
33. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.
34. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed elgamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95. Springer, 2020.
35. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.
36. Yoshikazu Hanatani, Yuichi Komano, Kazuo Ohta, and Noboru Kunihiro. Provably secure electronic cash based on blind multisignature schemes. In Giovanni Di Crescenzo and Aviel D. Rubin, editors, *Financial Cryptography and Data Security, 10th International Conference, FC 2006, Anguilla, British West Indies, February 27-March 2, 2006, Revised Selected Papers*, volume 4107 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 2006.
37. Lucjan Hanzlik, Yi-Fu Lai, Marzio Mula, Eugenio Paracucchi, Daniel Slamanig, and Gang Tang. Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 35–69. Springer, 2025.
38. Lucjan Hanzlik, Yi-Fu Lai, Eugenio Paracucchi, and Edoardo Persichetti. WOMBAT: Post-quantum blind signature from standard group action assumptions and more. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology –*

EUROCRYPT 2026, volume 16541 of *Lecture Notes in Computer Science*, pages 277–307. Springer, 2026.
39. Franklin Harding and Jiayu Xu. Unforgeability of blind schnorr in the limited concurrency setting. *IACR Commun. Cryptol.*, 1(3):16, 2024.
40. Julia Kastner, Julian Loss, and Jiayu Xu. On Pairing-Free Blind Signature Schemes in the Algebraic Group Model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2022.
41. Julia Kastner, Julian Loss, and Jiayu Xu. The Abe-Okamoto Partially Blind Signature Scheme Revisited. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 279–309. Springer, 2022.
42. Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI-Otter: Isogeny-Based (Partially) Blind Signatures from the Class Group Action with a Twist. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 729–761. Springer, 2023.
43. Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ROS: implication for isogeny and lattice-based blind signatures. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part I*, volume 14601 of *Lecture Notes in Computer Science*, pages 319–351. Springer, 2024.
44. Veronika Kuchta, Jason T LeGrow, and Edoardo Persichetti. Post-quantum (partially) blind signatures from matrix code equivalence. *Cryptography and Communications*, pages 1–53, 2026.
45. Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.
46. NIST. Post-Quantum Cryptography: Additional Digital Signature Schemes. https://csrc.nist.gov/projects/pqc-dig-sig, 2023. Accessed May 2025.
47. Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 109–118. ACM, 2011.
48. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, 2000.
49. Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptol. ePrint Arch.*, page 145, 2006.
50. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

51. Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *Information and Communications Security, Third International Conference, ICICS 2001, Xian, China, November 13-16, 2001*, volume 2229 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2001.
52. René Schoof. Elliptic curves over finite fields and the computation of square roots mod *p*. *Math. Comp.*, 44(170):483–494, 1985.
53. René Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7(1):219–254, 1995. Les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993).
54. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
55. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004.
56. Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986.
57. Andrew V. Sutherland. Identifying supersingular elliptic curves. *LMS J. Comput. Math.*, 15:317–325, 2012.
58. Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 582–612. Springer, 2022.
59. Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.
60. Xun Yi and Kwok-Yan Lam. A new blind ECDSA scheme for bitcoin transaction anonymity. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pages 613–620. ACM, 2019.