Key-Oblivious Encryption from isogenies and its application to Accountable Tracing Signatures.

Surbhi Shaw and Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur
Kharagpur-721302, India
surbhi_shaw@iitkgp.ac.in,ratna@maths.iitkgp.ac.in

Abstract. *Key-oblivious encryption* (KOE) is a newly developed cryptographic primitive that randomizes the public keys of an encryption scheme in an oblivious manner. It has applications in designing *accountable tracing signature* (ATS) that facilitates the group manager to revoke the anonymity of traceable users in a group signature while preserving the anonymity of non-traceable users. Despite of its importance and strong application, KOE has not received much attention in the literature.

In this work, we introduce the *first isogeny-based* KOE *scheme*. Isogeny is a fairly young post-quantum cryptographic field with sophisticated algebraic structures and unique security properties. Our KOE scheme is resistant to quantum attacks and derives its security from *Commutative Supersingular Decisional Diffie-Hellman* (CSSDDH), which is an isogeny based hard problem. More concretely, we have shown that our construction exhibits *key randomizability*, *plaintext indistinguishability under key randomization* and *key privacy under key randomization* in the standard model adapting the security framework of [KM15]. Furthermore, we have manifested instantiation of our scheme from cryptosystem based on *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH-512) [BKV19]. Additionally, we demonstrate the utility of our KOE scheme by leveraging it to construct an isogeny-based ATS scheme preserving anonymity under tracing, traceability, non-frameability, anonymity with accountability and trace obliviousness in the random oracle model following the security framework of [LNWX19].

Keywords. Post-quantum cryptography; Isogenies; Key-oblivious encryption; Accountable tracing signatures.

1 Introduction

Key-oblivious encryption (KOE), introduced by [KM15], is a promising cryptographic primitive which is being extensively studied recently [LNWX19]. The core concept in KOE is to enable randomization of a large set of public keys

related to the same secret key. This randomization generates related keys and the relation remains oblivious as long as the knowledge of the secret key and the randomness used are hidden.
Recent interest in designing a KOE is because of its application in developing a framework for *accountable tracing signature* (ATS). ATS is an enhanced variant of group signature. In the traditional group signature scheme, the group manager (GM) is allowed to randomly revoke the anonymity of any signer in order to avoid the misuse of anonymity of the signer. The GM is trusted blindly and there is no means to check his accountability. On the other hand, the GM is kept accountable for his actions in an ATS scheme. Once a user enrolls in the group, the GM determines the category of the user. The traceable users are the suspected users and their anonymity can be revoked by the GM. For non-traceable users, anonymity remains preserved and even the GM cannot trace the signatures generated by them. The GM then issues a certificate corresponding to his choice (traceable/non-traceable) to the user. Later the group GM reveals his choice of category in order to enforce his accountability.
How is KOE different from PKE? KOE is nontrivial and useful particularly when key privacy is at prime concern apart from data privacy. The traditional security prerequisite of any *public key encryption* (PKE) scheme is to provide privacy of the encrypted data only. KOE captures this data privacy requirement by the security attribute *plaintext indistinguishability under key randomization.* Besides data privacy, KOE seeks to provide two different security requirements of an encryption scheme, which is a variant of standard key privacy and is formalized by *key randomizability* and *key privacy under key randomization.* Key randomizability requires that an adversary cannot distinguish between the original public key and a randomized public key without having the secret key. In contrast, key privacy under key randomization requires anonymity from the adversary's point of view. An adversary in possession of a ciphertext is unable to tell which particular key from a set of adversarially randomized public keys is used to create the ciphertext. There exist encryption schemes that are able to meet *indistinguishability under chosen-ciphertext attack* (IND-CCA), which is the most potent form of data privacy, but do not provide key privacy. Designing KOE with the above stated three security requirements is a challenging task.
Our Contributions. The only two existing KOE constructions so far are [KM15] and [LNWX19]. Considering the limited development in the area of KOE, we concentrate on designing KOE from isogenies that withstands quantum attacks. The closest related work to ours is that of [KM15]. The KOE scheme presented in [KM15] relies on the *Decisional Diffie-Hellman* (DDH) assumption and does not provide security in the quantum world due to Shor's algorithm [Sho99]. San Ling et al. [LNWX19] introduced KOE in the lattice settings which is secure under the hardness of *Ring Learning With Errors* (RLWE) assumption. Our contribution in this paper is threefold and can be summed up as follows:

– *Firstly*, we initiate the study of KOE in the isogeny world. We have developed the first isogeny-based KOE and named it as *Commutative Supersingular Isogeny Key-Oblivious Encryption* (CSIKOE). We provide a concrete security analysis and have shown that our scheme satisfies *key randomizability*, *plaintext indistinguishability under key randomization* and *key privacy under key randomization* in the standard model. (Section 3)

– *Secondly*, we have manifested instantiation of our CSIKOE scheme from cryptosystem based on *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH-512) parameter set. We refer to it as CSIKOE-512. (Section 4)

– *Finally*, we exploit our CSIKOE-512 scheme to develop an isogeny-based ATS scheme to make the group manager accountable in a group signature scheme. (Section 5)

In 2019, Castryck et al. gave the non-interactive key exchange based on isogeny, named as CSIDH [CLM$^+$18]. The ElGamal-like PKE based on CSIDH without using hash functions is not *indistinguishability under chosen-plaintext attack* (IND-CPA) secure [MOT20]. Thus, we construct our CSIKOE scheme leveraging the Hashed-PKE (Section 2.2) based on CSIDH which is IND-CPA secure. We believe our CSIKOE scheme is efficient in terms of storage and communication cost. For a security parameter λ, our CSIKOE scheme features user public key, user secret key and ciphertext size of $O(\lambda)$ each. Our CSIKOE can be instantiated with any of the three sets of CSIDH parameters that have been introduced till now (CSIDH-512, CSIDH-1024, and CSIDH-1792). However, the class group structure of the quadratic imaginary field corresponding to the CSIDH-512 parameter set is only known [BKV19]. We emphasize that our CSIKOE scheme derived from the CSIDH-512 parameter set turns out to be more efficient. We provide a detailed security analysis and arrived at the following result:

Theorem 1. *Under the CSSDDH assumption, our isogeny-based CSIKOE scheme satisfies key randomizability, plaintext indistinguishability under key randomization and key privacy under key randomization in the standard security model following the security framework of [KM15].*

We then exhibit an application of our KOE scheme by developing the first ATS scheme from isogenies. We integrate the *Commutative Supersingular Isogeny based Fiat-Shamir* (CSI-FiSh) signature scheme [BKV19] (Section 2.4) and a zero-knowledge argument system (Section 2.3) in our CSIKOE-512 in a suitable manner to yield an ATS scheme. We have arrived at the following theorem:

Theorem 2. *Under the assumption that CSI-FiSh signature scheme is strongly unforgeable, CSIKOE-512 scheme satisfies key randomizability, plaintext indistinguishability under key randomization and key privacy under key randomization and Π is zero-knowledge simulation-extractable argument system, our isogeny based ATS scheme satisfies anonymity under tracing, traceability, non-frameability, anonymity with accountability and trace obliviousness in the random oracle model following the security framework of [LNWX19].*

2 Preliminaries

Notation. Throughout this paper, we use the following notations: Let $\lambda \in \mathbb{N}$ denote the security parameter. We use #S to denote the cardinality of the set S. The residue class ring is denoted by $\mathbb{Z}/q\mathbb{Z}$. The symbol ' $\cong$ ' is used to denote isomorphism. A function $\mu(\cdot)$ is negligible if for every integer c, there exists an integer k such that for all $\lambda > k$, $|\mu(\lambda)| < 1/\lambda^c$. The sign function sgn is defined as $\mathsf{sgn}(x) = 1$ if $x > 0$, $\mathsf{sgn}(x) = -1$ if $x < 0$ and $\mathsf{sgn}(x) = 0$ for $x = 0$. By $\mathsf{bin}(x)$, we mean the binary representation of the argument x. For two strings s_1, s_2, $s_1||s_2$ represents the concatenation of strings s_1 and s_2.

Elliptic curves and isogenies. Let K be a finite field and $\overline{K}$ be its algebraic closure. An elliptic curve E over K is a non-singular, projective, cubic curve having genus one with a special point O, called the point at infinity. The set of K-rational points of the elliptic curve E forms an addition abelian group with O as the identity element.

Definition 1. *(Montgomery curve). An elliptic curve E defined over a finite field K is called a Montgomery curve if it satisfies $E : By^2 = x^3 + Ax^2 + x$ where $B(A^2 - 4) \neq 0$ for some $A, B \in K$.*

Definition 2. *(Isogeny). Let E_1 and E_2 be two elliptic curves over a finite field K. An isogeny from E_1 to E_2 is a non-constant morphism $\phi : E_1 \longrightarrow E_2$ over $\overline{K}$, preserving the point at infinity O. Two elliptic curves E_1 and E_2 are isogenous if there exists an isogeny from E_1 to E_2.*

Example 1. The most common and well-known examples of isogeny includes:
(i) For every elliptic curve E and for each $m \in \mathbb{Z}$, the multiplication-by-m map, $[m]$: $E \longrightarrow E$ is an isogeny defined by: $[m](P) = P + P + \cdots + P$ (m times) if $m > 0$; $[m](P) = [-m](-P)$ if $m < 0$; and $[0](P) = O$.
(ii) Let E be an elliptic curve over $\mathbb{F}_p$. Then the Frobenius endomorphism, denoted by π, which maps (x, y) to (x^p, y^p) is an isogeny.

Theorem 3. *([Sil09]). Let $\phi : E_1 \longrightarrow E_2$ be an isogeny. Then $\phi(P + Q) = \phi(P) + \phi(Q)$ for all $P, Q \in E_1$. In other words, ϕ preserves the group structure of E_1 and is a group homomorphism.*

Definition 3. *([Sil09]) Let E_1 and E_2 be two curves over K and $\phi : E_1 \longrightarrow E_2$ be an isogeny.*

(Degree of an isogeny). The degree of the isogeny ϕ, denoted by $\mathsf{deg}(\phi)$ is defined to be the degree of the field extension $\overline{K}(E_1)/\phi^*\overline{K}(E_2)$ where ϕ^* is induced by the isogeny ϕ and is the injection of the function fields fixing the field K:

$$\phi^* : \overline{K}(E_2) \hookrightarrow \overline{K}(E_1)$$
$$\phi^* : f \longrightarrow f \circ \phi$$

(Kernel of an isogeny). The kernel of the isogeny ϕ, denoted by $\mathsf{ker}(\phi)$ *is defined as:* $\mathsf{ker}(\phi) = \{P \in E_1(\overline{K}) : \phi(P) = O\}$.

(Separable isogeny). The isogeny ϕ is called a separable isogeny if and only if $\#\mathsf{ker}(\phi) = \mathsf{deg}(\phi)$.

Definition 4. *(Endomorphism ring). The set of all isogenies from the elliptic curve E to itself defined over $\overline{K}$ forms a ring under pointwise addition and composition. This ring is called an endomorphism ring of the elliptic curve E and is denoted by* $\mathsf{End}(E)$. *By* $\mathsf{End}_K(E)$, *we mean the set of all isogenies from E to itself defined over K.*

If the endomorphism ring $\mathsf{End}(E)$ of an elliptic curve E is isomorphic to an order in a quaternion algebra, the elliptic curve E is said to be *supersingular*. On the other hand, if $\mathsf{End}(E)$ is isomorphic to an order in an imaginary quadratic field, we say the elliptic curve E is *ordinary*.

Theorem 4. *[Wat69] Let E_1 be an elliptic curve and G be a finite subgroup of E_1. Then there is a unique elliptic curve E_2 and a separable isogeny $\phi : E_1 \longrightarrow E_2$ with* $\mathsf{ker}(\phi) = G$ *such that $E_2 \cong E_1/G$.*

Given a subgroup G of E_1, one can explicitly acquire an isogeny $\phi : E_1 \to E_2$ with $\mathsf{ker}(\phi) = G$ satisfying $E_2 \cong E_1/G$ leveraging Vélu's formulae [Vél71].

Definition 5. *Given a group G with identity element e and a set S, a group action $*$ of G on S is a function $* : G \times S \to S$, satisfying the following two axioms:*
(Identity): $e * s = s$ for all $s \in S$
(Compatibility): $g * (g' * s) = (gg') * s$ for all g and g' in G and all $s \in S$.

Hard homogeneous spaces. Hard homogeneous spaces [Cou06] demand the following problems to be computationally easy:

(Group Operation): Given two strings g and g' decide whether they represent elements of G, check whether $g = g'$, compute g^{-1} and gg'.
(Random element): Sample a uniformly random element from the group G.
(Membership): Decide whether a string s represents an element in the set S.
(Equality): Given two elements of the set S, check their equality.
(Action): Compute $g * s$, i.e., the action of a group element $g \in G$ on some element $s \in S$.
Besides, there should not be any polynomial-time solver for the following problems:
(Vectorization): Given s, $s' \in S$, find $g \in G$ such that $g * s = s'$.
(Parallelization): Given $s_0, s'_0, s_1 \in S$ such that $g * s_0 = s'_0$ for some $g \in G$, find $s'_1 = g * s_1$.

The discrete logarithm problem and the traditional Diffie-Hellman problem on general groups are particular instances of HHS. The discrete logarithm problem in S is the same as the vectorization problem and the computational Diffie–Hellman problem is similar to the parallelization problem.

Definition 6. *(Ideal class groups [MOT20]). Let $\mathcal{O}$ be an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$.*

(Fractional ideal). A fractional ideal of $\mathcal{O}$ is an $\mathcal{O}$-sub module of a field K of the form $\alpha\mathfrak{a}$ where $\alpha \in K^* = K \setminus \{0\}$ and $\mathfrak{a}$ is an $\mathcal{O}$-ideal.
(Invertible fractional ideal). An invertible fractional ideal $\mathfrak{a}$ of $\mathcal{O}$ is defined as a fractional ideal of $\mathcal{O}$ that satisfies $\mathfrak{a}\mathfrak{b} = \mathcal{O}$ for some fractional ideal $\mathfrak{b}$ of $\mathcal{O}$. The ideal $\mathfrak{b}$ can be represented as $\mathfrak{a}^{-1}$.
(Ideal class group). Let $\mathcal{I}(\mathcal{O})$ be a set of invertible fractional ideals of $\mathcal{O}$. Then $\mathcal{I}(\mathcal{O})$ is an abelian group derived from the multiplication of ideals with the identity $\mathcal{O}$. Let $\mathcal{P}(\mathcal{O})$ be a subgroup of $\mathcal{I}(\mathcal{O})$ defined by $\mathcal{P}(\mathcal{O}) = \{\ \mathfrak{a}\ /\ \mathfrak{a} = \alpha\mathcal{O}$ for some $\alpha \in K^*\ \}$. The abelian group $\mathsf{Cl}(\mathcal{O})$ defined by $\mathcal{I}(\mathcal{O})/\mathcal{P}(\mathcal{O})$ is said to be the ideal class group of $\mathcal{O}$. An element of $\mathsf{Cl}(\mathcal{O})$ is an equivalence class of $\mathfrak{a}$, denoted by $[\mathfrak{a}]$.

Let $\mathsf{Ell}_p(\mathcal{O})$ denote the set of $\mathbb{F}_p$-isomorphic classes of supersingular elliptic curves E, whose $\mathbb{F}_p$-endomorphism ring $\mathsf{End}_{\mathbb{F}_p}(E) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. The action of the ideal class group $G = \mathsf{Cl}(\mathcal{O})$ on the set $S = \mathsf{Ell}_p(\mathcal{O})$ is computed as in Algorithm 1.

Algorithm 1: Computing class group action

```
  Input  : $[\mathfrak{a}] \in \mathsf{Cl}(\mathcal{O})$, $E \in \mathsf{Ell}_p(\mathcal{O})$.
  Output: $[\mathfrak{a}] * E$.
1 Let $\mathfrak{a}$ be the integral representative of the ideal class $[\mathfrak{a}]$.
2 Computes the subgroup $E[\mathfrak{a}] = \bigcap_{\alpha\in\mathfrak{a}} \mathsf{ker}(\alpha)$.
3 Computes an elliptic curve $E/E[\mathfrak{a}]$ and an isogeny $\phi_{\mathfrak{a}} : E \longrightarrow E/E[\mathfrak{a}]$
    using Velu's formula.                                    // See theorem 4
4 return $E/E[\mathfrak{a}]$.
```

Henceforth, we shall use the notation $[\mathfrak{a}]E$ instead of $[\mathfrak{a}] * E$ to denote the curve $E/E[\mathfrak{a}]$ obtained by the action of class group element $[\mathfrak{a}] \in \mathsf{Cl}(\mathcal{O})$ on the elliptic curve $E \in \mathsf{Ell}_p(\mathcal{O})$ where $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Theorem 5. *([Wat69]) Let $\mathcal{O}$ be an order of an imaginary quadratic field and E be an elliptic curve defined over $\mathbb{F}_p$. If $\mathsf{Ell}_p(\mathcal{O})$ contains the $\mathbb{F}_p$-isomorphism class of supersingular elliptic curves, then the action of the ideal class group $\mathsf{Cl}(\mathcal{O})$ on $\mathsf{Ell}_p(\mathcal{O})$, defined by*

$$\mathsf{Cl}(\mathcal{O}) \times \ \mathsf{Ell}_p(\mathcal{O}) \longrightarrow \mathsf{Ell}_p(\mathcal{O})$$
$$([\mathfrak{a}], E) \longrightarrow E/E[\mathfrak{a}]$$

is free and transitive where $\mathfrak{a}$ *is an integral ideal of* $\mathcal{O}$ *and* $E[\mathfrak{a}]$ *is the intersection of the kernels of elements in* $\mathfrak{a}$.

Theorem 6. *[CLM+18] Let* $p \geq 5$ *be a prime such that* $p \equiv 3 \pmod 8$ *and* E *be a supersingular elliptic curve over* $\mathbb{F}_p$. *Then* $\mathsf{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[\sqrt{-p}]$ *if and only if there exists* $A_{\mathsf{mg}} \in \mathbb{F}_p$ *such that* E *is* $\mathbb{F}_p$*-isomorphic to the Montgomery elliptic curve* $E_{A_{\mathsf{mg}}} : y^2 = x^3 + A_{\mathsf{mg}}x^2 + x$. *Moreover, if such an* A_{mg} *exists then it is unique.*

2.1 CSIDH: a non-interactive key exchange based on isogeny

The non-interactive key exchange scheme CSIDH = (Setup, KeyGen, Key Exchange) [CLM+18] consists of three polynomial-time algorithms satisfying the following requirements:
Setup(1^λ)$\rightarrow$ pp : A trusted authority runs this probabilistic polynomial-time (PPT) algorithm on input a security parameter λ and performs the following steps:

– Chooses a large prime p of the form $p = 4\,l_1 l_2 \ldots l_n - 1$, where the l_i are small distinct odd primes.

– Picks an integer m and selects a base elliptic curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

– Sets the public parameter $\mathsf{pp} = (p,\, l_1,\, l_2,\, \ldots,\, l_n,\, m,\, E_0)$.

KeyGen(pp) $\rightarrow$ (pk, sk): A user, say U on input the public parameter pp runs this randomized algorithm by executing the following steps and generate its public key pk and secret key sk.

– Samples a vector $\mathsf{u} = (u_1, u_2, \ldots, u_n)$ of integers randomly where $u_i \in$ [-m, m], $i = 1, 2, \ldots, n$ and defines $[\mathfrak{u}] \in \mathsf{Cl}(\mathcal{O})$ as $[\mathfrak{u}] = [\mathfrak{l}_1^{u_1} \mathfrak{l}_2^{u_2} \ldots \mathfrak{l}_n^{u_n}]$, where $\mathfrak{l}_i$ $= < l_i, \pi - 1 >$. Here the notation $<,>$ denotes the ideal generated by multiplication by l_i map and $\pi - 1$ where π is the Frobenius endomorphism.

– Computes the action of $[\mathfrak{u}] \in \mathsf{Cl}(\mathcal{O})$ on $E_0 \in \mathsf{Ell}_p(\mathcal{O})$ to get the curve $[\mathfrak{u}]E_0$. Computes the unique Montgomery coefficient $U_{\mathsf{mg}} \in \mathbb{F}_p$ of the elliptic curve $[\mathfrak{u}]E_0 : y^2 = x^3 + U_{\mathsf{mg}}x^2 + x$ formed by the action of $[\mathfrak{u}]$ on E_0. (See Remark 1)

– Sets its public key $\mathsf{pk} = U_{\mathsf{mg}}$ and secret key $\mathsf{sk} = \mathsf{u}$.

Key Exchange: Suppose users A and B want to agree upon a common secret. Let user A is having her public-secret key pair $(\mathsf{pk}_A,\, \mathsf{sk}_A) \leftarrow$ KeyGen(pp) and user B is having his public-secret key pair $(\mathsf{pk}_B,\, \mathsf{sk}_B) \leftarrow$ KeyGen(pp). User A has the secret key $\mathsf{sk}_A = \mathsf{a}$ and public key $\mathsf{pk}_A = A_{\mathsf{mg}}$, where $A_{\mathsf{mg}} \in \mathbb{F}_p$ is the Montgomery coefficient of the elliptic curve $[\mathfrak{a}]E_0 : y^2 = x^3 + A_{\mathsf{mg}}x^2 + x$. User B has the secret key $\mathsf{sk}_B = \mathsf{b}$ and public key $\mathsf{pk}_B = B_{\mathsf{mg}}$, where $B_{\mathsf{mg}} \in \mathbb{F}_p$ is the Montgomery coefficient of the curve $[\mathfrak{b}]E_0 : y^2 = x^3 + B_{\mathsf{mg}}x^2 + x$.

– User A uses $\mathsf{pk}_B = B_{\mathsf{mg}}$ to form the elliptic curve $[\mathfrak{b}]E_0 : y^2 = x^3 + B_{\mathsf{mg}}x^2 + x$, applies the action of $[\mathfrak{a}]$ to $[\mathfrak{b}]E_0$ and computes the elliptic curve $[\mathfrak{a}][\mathfrak{b}]E_0$.

– Similarly, user B proceeds with his own secret $\mathsf{sk}_B = \mathsf{b}$, uses $\mathsf{pk}_A = A_{\mathsf{mg}}$ and computes the curve $[\mathfrak{b}][\mathfrak{a}]E_0$.

– The Montgomery coefficient that uniquely identifies the common secret curve $[\mathfrak{a}][\mathfrak{b}]E_0 = [\mathfrak{b}][\mathfrak{a}]E_0$ serves as the shared key between user A and user B.

Remark 1. We dwell upon few important aspects of CSIDH.

1. Here the prime $p = 4\,l_1 l_2 \dots l_n - 1$ where l_i are small distinct odd primes is designed such that $p \equiv 3 \pmod 4$ and $p \geq 5$ in order to use Montgomery coefficient of curves using Theorem 6. As the elliptic curve $[\mathfrak{u}]E_0$ is a member of $\mathsf{Ell}_p(\mathcal{O})$, we have $\mathsf{End}_{\mathbb{F}_p}([\mathfrak{u}]E_0) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. Theorem 6 guarantees that there exists a unique $U_{\mathsf{mg}} \in \mathbb{F}_p$ such that $[\mathfrak{u}]E_0$ is $\mathbb{F}_p$-isomorphic to the Montgomery elliptic curve $E_{U_{\mathsf{mg}}} : y^2 = x^3 + U_{\mathsf{mg}}x^2 + x$.
2. As the cardinality of the class group is asymptotically $\#\mathsf{Cl}(\mathcal{O}) \sim \sqrt{\Delta}$, thus it is computationally infeasible to compute the structure of the class group $\mathsf{Cl}(\mathcal{O})$, where Δ stands for discriminant of class group. Thus [CLM+18] opts for heuristics arguments assuming that $\mathfrak{l}_i$ do not have very small order and are uniformly distributed in the class group, two ideals $\mathfrak{l}_1^{a_1}\mathfrak{l}_2^{a_2}\cdots\mathfrak{l}_n^{a_n}$ for small a_i will occasionally lie in the same class group. The exponents a_i's are preferred to be sampled from a short range $\{-m, \cdots, m\}$ for some integer m such that $2m+1 \geq \sqrt[n]{\#\mathsf{Cl}(\mathcal{O})}$.
3. Choosing prime p of the form $4\,l_1 l_2 \dots l_n - 1$, establishes an association of the fractional ideal $\mathfrak{l}_i =< l_i, \pi - 1 >$ to each l_i. The action of these $\mathfrak{l}_i$ can be computed efficiently by finding an $\mathbb{F}_p$-rational point and hence a unique subgroup of $E_0(\mathbb{F}_p)$ of order l_i and applying Velu's formulas [Vél71].

Correctness. Correctness of CSIDH follows immediately from the commutativity of the class group $\mathsf{Cl}(\mathcal{O})$ and Theorem 6.

Theorem 7. *The non-interactive key exchange protocol CSIDH is secure under the Commutative Supersingular Decisional Diffie-Hellman (CSSDDH) assumption as defined in Definition 7.*

Definition 7. *Let p be a large prime of the form $p = 4\,l_1 l_2 \dots l_n - 1$, where the l_i are small distinct odd primes and E_0 be the base elliptic curve given by $y^2 = x^3 + x$ over $\mathbb{F}_p$. Let $[\mathfrak{a}]$, $[\mathfrak{b}]$, $[\mathfrak{c}]$ be random element of $Cl(\mathcal{O})$, where $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. Let λ be the bit length of p. The Commutative Supersingular Decisional Diffie-Hellman (CSSDDH) advantage of any PPT adversary denoted by $Adv_{\mathcal{A}}^{CSSDDH}(\lambda)$ is defined as:*

$$\begin{aligned} Adv_{\mathcal{A}}^{CSSDDH}(\lambda) = {} & |Pr[\mathfrak{a}, \mathfrak{b} \leftarrow Cl(\mathcal{O}) \mid \mathcal{A}(\, E_0,\, [\mathfrak{a}]E_0,\, [\mathfrak{b}]E_0,\, [\mathfrak{a}][\mathfrak{b}]E_0\,) = 1] \\ & - Pr[\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \leftarrow Cl(\mathcal{O}) \mid \mathcal{A}(\, E_0,\, [\mathfrak{a}]E_0,\, [\mathfrak{b}]E_0,\, [\mathfrak{c}]E_0\,) = 1]\,|. \end{aligned}$$

We say that the CSSDDH assumption holds if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CSSDDH}}(\lambda)$ of any PPT adversary is negligible.

2.2 Hashed-PKE: a hash-based public key encryption from CSIDH

We explain below $\mathsf{Hashed\text{-}PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, an Elgamal-like PKE based on CSIDH using hash functions.
$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: A trusted authority runs this algorithm on input a security parameter λ and performs the following steps:

- Chooses a large prime p of the form $p = 4\,l_1 l_2 \dots l_n - 1$, where the l_i are small distinct odd primes.
- Picks an interger m such that $2m+1 \geq \sqrt[n]{\#\mathsf{Cl}(\mathcal{O})}$ and selects a base elliptic curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.
- Defines $M_C : \mathsf{Ell}_p(\mathcal{O}) \to \mathbb{F}_p$, a function that maps isomorphism classes of elliptic curve to its Montgomery coefficient.
- Samples a family of keyed "hash" function $\mathcal{H} := \{H_k\}_{k \in K}$ where $H_k : \mathbb{F}_p \to \{0,1\}^\lambda$ for each $k \in K$, where K is the key space and let the message space be $\mathcal{M} = \{0,1\}^\lambda$.
- Sets the public parameter $\mathsf{pp} = (p, l_1, l_2, \dots, l_n, m, E_0, M_C, \mathcal{H} := \{H_k\}_{k \in K})$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Any user U can run this randomized algorithm on input the public parameter pp and generate its public key pk and secret sey sk as follows:

- Samples a vector $\mathsf{u} = (u_1, u_2, \dots, u_n)$ of integers randomly where $u_i \in$ [-m, m], $i = 1, 2, \dots, n$.
- Defines $[\mathfrak{u}] \in \mathsf{Cl}(\mathcal{O})$ as $[\mathfrak{u}] = [\mathfrak{l}_1^{u_1} \mathfrak{l}_2^{u_2} \dots \mathfrak{l}_n^{u_n}]$, where $\mathfrak{l}_\mathsf{i} = < l_i, \pi - 1 >$. Here the notation $< , >$ denotes the ideal generated by multiplication by l_i map and $\pi - 1$ where π is the Frobenius endomorphism.
- Computes the curve $E_U = [\mathfrak{u}]E_0$ and sets her public key $\mathsf{pk} = E_U$ and secret key $\mathsf{sk} = \mathsf{u}$.

$\mathsf{Enc}(\mathsf{pp}, m, \mathsf{pk}) \to \mathsf{ct}$: An encryptor B runs this algorithm to encrypt plaintext $m \in \mathcal{M}$ using public parameter pp and recipient A′s public key $\mathsf{pk} = \mathsf{pk}_A = E_A$ in the following manner.

- Samples a random integer vector $\mathsf{b} = (b_1, b_2, \dots, , b_n)$ where $b_i \in$ [-m, m] for $i = 1, 2, \dots n$. This defines an element $[\mathfrak{b}] = [\mathfrak{l}_1^{b_1} \mathfrak{l}_2^{b_2} \dots \mathfrak{l}_n^{b_n}] \in \mathsf{Cl}(\mathcal{O})$ where $\mathfrak{l}_\mathsf{i} = < l_i, \pi - 1 >$.
- Computes $[\mathfrak{b}]E_0$, $[\mathfrak{b}]E_A$ and returns ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ where $\mathsf{ct}_1 = [\mathfrak{b}]E_0$ and $\mathsf{ct}_2 = m \oplus H_k(M_C([\mathfrak{b}]E_A))$.

$\mathsf{Dec}(\mathsf{pp},\,\mathsf{ct},\,\mathsf{sk}) \to m$: The decryptor A runs this deterministic algorithm taking input the public parameter pp, the ciphertext $\mathsf{ct} = (\mathsf{ct}_1,\,\mathsf{ct}_2)$ and her own secret key $\mathsf{sk} = \mathsf{sk}_A = \mathsf{a}$. She retrieves the message m by computing the action of $[\mathfrak{a}]$ on $\mathsf{ct}_1 = [\mathfrak{b}]E_0$ to produce the elliptic curve $[\mathfrak{a}]\mathsf{ct}_1$ and then computing $\mathsf{ct}_2 \oplus H_k(M_C([\mathfrak{a}]\mathsf{ct}_1))$.

Correctness: We say that $\mathsf{Hashed\text{-}PKE}$ is correct if for all security parameter λ, all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(\mathsf{pk},\,\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, all m, it must hold that $\mathsf{Dec}(\mathsf{Enc}(\mathsf{pp},\,m,\,\mathsf{pk}),\,\mathsf{pp},\,\mathsf{sk}) = m$.

Note that the ciphertext in the protocol $\mathsf{Hashed\text{-}PKE}$ is given by: $\mathsf{ct} = (\mathsf{ct}_1 = E_B,\,\mathsf{ct}_2 = m \oplus H_k(M_C([\mathfrak{b}]E_A)))$ and consequently we have,

$$\begin{aligned}\mathsf{ct}_2 \oplus H_k(M_C([\mathfrak{a}]\mathsf{ct}_1)) &= m \oplus H_k(M_C([\mathfrak{b}]E_A))) \oplus H_k(M_C([\mathfrak{a}]\mathsf{ct}_1))\\ &= m \oplus H_k(M_C([\mathfrak{b}][\mathfrak{a}]E_0))) \oplus H_k(M_C([\mathfrak{a}][\mathfrak{b}]E_0))\\ &= m\end{aligned}$$

Theorem 8. *The $\mathsf{Hashed\text{-}PKE}$ scheme is indistinguishable under chosen plaintext attack (IND-CPA) under the CSSDDH assumption given in Definition 7.*

Definition 8. *Let $\mathcal{H} := \{H_k\}_{k\in K}$ be a family of keyed hash functions where each H_k is a function that maps from group G to $\{0,1\}^l$, where l denotes the length of the string. Let $\mathcal{A}$ be an algorithm that takes as input an element of key space K and an element of $\{0,1\}^l$, and outputs a bit. We define the entropy smoothing advantage $\mathsf{Adv}^{\mathsf{ES}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ to be:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ES}}_{\mathcal{A}}(\lambda) = |\mathsf{Pr}[\,&k \leftarrow K, g \leftarrow G \,|\, \mathcal{A}(k,\,H_k(g)) = 1\,]\\ &-\ \mathsf{Pr}[\,k \leftarrow K, h \leftarrow \{0,1\}^l \,|\, \mathcal{A}(k,\,h) = 1\,]\,|.\end{aligned}$$

We say that the family of keyed hash functions $\mathcal{H} := \{H_k\}_{k\in K}$ is entropy smoothing if the entropy smoothing advantage $\mathsf{Adv}^{\mathsf{ES}}_{\mathcal{A}}(\lambda)$ of any PPT algorithm $\mathcal{A}$ is negligible.

2.3 Non-Interactive Zero-Knowledge

Definition 9. *A Non-Interactive Zero-Knowledge (NIZK) argument system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2))$ for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$ specifies the following PPT algorithms:*

$\mathsf{Setup}(1^\lambda) \to crs$: A trusted party runs this randomized algorithm taking input the security parameter λ and generates a common reference string crs which is accessible to everyone.

$\mathsf{Prove}(crs,\,x,\,w) \to \pi$: To prove the statement $x \in \mathcal{L}$ with witness w, the prover runs this randomized algorithm taking the crs and generates a proof π.

$\mathsf{Verify}(crs,\,x,\,\pi) \to \{0,1\}$: This is a deterministic algorithm run by a verifier that takes input the crs, a statement x and a proof π and returns 1 if the proof π is valid, else returns 0.

A non-interactive zero-knowledge argument system has the following three requirements: *Completeness*, *Soundness* and *Zero-Knowledge*.

Definition 10. *(Completeness). A non-interactive zero-knowledge argument system Π for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$ is complete if for all x, w such that $\mathcal{R}(x, w) = 1$ and for all strings $crs \leftarrow \mathsf{Setup}(1^\lambda)$, it must hold that $\mathsf{Verify}(crs, x, \mathsf{Prove}(crs, x, w)) = 1$.*

Definition 11. *(Soundness). A non-interactive zero-knowledge argument system Π for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$ is sound if the soundness advantage $\mathsf{Adv}^{\mathsf{SnD}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ of any PPT adversary $\mathcal{A}$ given by $\mathsf{Adv}^{\mathsf{SnD}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{SnD}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = 1]$ is negligible where the experiment $\mathsf{Exp}^{\mathsf{SnD}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ is specified in Fig. 1.*

Definition 12. *(Zero-knowledge). A non-interactive zero-knowledge argument system Π for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$ is zero-knowledge if the zero-knowledge advantage $\mathsf{Adv}^{\mathsf{ZoK}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ of any PPT adversary $\mathcal{A}$ given by $\mathsf{Adv}^{\mathsf{ZoK}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{ZoK}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = 1]$ is negligible where $\mathcal{S}_1, \mathcal{S}_2$ in the experiment $\mathsf{Exp}^{\mathsf{ZoK}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ as specified in Fig. 1 stands for the simulator and trap refers to the trapdoor.*

Definition 13. *(NIZK argument of knowledge). A non-interactive zero-knowledge argument system Π for a language $\mathcal{L} \in \mathsf{NP}$ with witness relation $\mathcal{R}$ is an argument of knowledge if there exists a PPT extractor $\mathsf{E} = (\mathsf{E}_1, \mathsf{E}_2)$ such that the extracting advantage $\mathsf{Adv}^{\mathsf{ExT}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ of any PPT adversary $\mathcal{A}$ given by $\mathsf{Adv}^{\mathsf{ExT}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{ExT}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) = 1]$ is negligible, where trap refers to a trapdoor in the experiment $\mathsf{Exp}^{\mathsf{ExT}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ as specified in Fig. 1.*

```
$\mathsf{Exp}^{\mathsf{SnD}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}(1^\lambda)$
$(x, \pi) \leftarrow \mathcal{A}(crs)$
if $(x \notin \mathcal{L} \wedge \mathsf{Verify}(crs, x, \pi))$
  return 1
else return 0.

$\mathsf{Exp}^{\mathsf{ExT}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}(1^\lambda)$
$(crs, trap) \leftarrow \mathsf{E}_1(1^\lambda)$
$(x, \pi) \leftarrow \mathcal{A}(crs)$
$w \leftarrow \mathsf{E}_2(crs, trap, x, \pi)$
if $(\mathcal{R}(x, w) = 0 \wedge \mathsf{Verify}(crs, x, \pi))$
  return 1
else return 0.

$\mathsf{Exp}^{\mathsf{ZoK}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$

$crs_1 \leftarrow \mathsf{Setup}(1^\lambda)$
$(crs_0, trap) \leftarrow \mathcal{S}_1(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{Prove}}(crs_b)$
if $(b = b')$
  return 1
else return 0.

Oracle $\mathsf{Prove}(x, w)$

if $\mathcal{R}(x, w) = 0$, return $\perp$
if $b = 1$ then $\pi \leftarrow \mathsf{Prove}(crs_1, x, w)$
else $\pi \leftarrow \mathcal{S}_2(crs_0, x, trap)$
return $\pi$.
```

Fig. 1. Experiment defining soundness, zero-knowledge, NIZK argument of knowledge

2.4 CSI-FiSh : a signature scheme based on isogeny

The *Commutative Supersingular Isogeny based Fiat-Shamir signature* [BKV19] $\mathsf{CSI\text{-}FiSh} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of four polynomial-time algorithms with the following requirements:

$\mathsf{Setup}(1^\lambda) \longrightarrow \mathsf{pp}$: Setup algorithm is same as that of setup algorithm of CSIKOE-512 except the hash function. Here the trusted party samples a cryptographic hash function $H' : \{0,1\}^* \to [-S,\, S]^t$ where $t = \lambda/\log S$ and sets the public parameter $\mathsf{pp} = (p,\, \mathfrak{g},\, N,\, E_0, H')$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$: This is a randomized algorithm run by a user that takes input the public parameter pp and generates a signing key sk and verification key vk in the following manner:

- Samples $S-1$ elements $[\mathfrak{a}_i] = [\mathfrak{g}^{a_i}] \in \mathcal{G}$ for some $a_i \in \mathbb{Z}_N$ for $i = 1, \ldots, S-1$.
- Computes the elliptic curve $E_i = [a_i]E_0$ for $i = 1, 2, \ldots, S-1$.
- Sets the signing key $\mathsf{sk} = (a_1, a_2, \ldots, a_{S-1})$ and verification key $\mathsf{vk} = \{E_1,\, E_2, \ldots, E_{S-1}\}$.
- Publishes verification key vk and keeps sk secret to himself.

$\mathsf{Sign}(\mathsf{sk},\, msg) \to \sigma$: This randomized algorithm is run by a signer that generates a signature σ on the message msg using his signing key sk in the following manner:

- Sets $a_0 \leftarrow 0$ and samples $[\mathfrak{a}'_i] = [\mathfrak{g}^{a'_i}] \in \mathcal{G}$ for some $a'_i \in \mathbb{Z}_N$ for $i = 1, 2, \ldots, t$.
- Computes t many commitment elliptic curves $E^{(i)} = [a'_i]E_0$ for $i = 1, 2, \ldots, t$.
- Computes the challenge bits $(ch_1,\, ch_2, \ldots, ch_t) = H'(E^{(1)}||\ldots||E^{(t)}||msg)$.
- Computes the response $z_i = a'_i$ - $\text{sign}(ch_i)\, a_{|\, ch_i\, |} \pmod N$ for $i = 1, 2, \ldots, t$.
- Sets the signature $\sigma = (ch_1, ch_2, \ldots, ch_t, z_1, z_2, \ldots, z_t)$.

$\mathsf{Verify}(\mathsf{vk},\, msg,\, \sigma) \to \{0,1\}$: This is a deterministic algorithm run by a verifier that checks the validation of the signature σ on the message msg using his verification key vk.

- Parse $\sigma = (ch_1, ch_2, \ldots, ch_t, z_1, z_2, \ldots, z_t)$.
- Define $E_{-i} = E_i^t$ for $i = 1, 2, \ldots, S-1$ where E_i^t is the twist of the elliptic curve E_i.
- Computes t many elliptic curves $E^{(i)} = [z_i]E_{ch_i}$ for $i = 1, 2, \ldots, t$.
- Computes $(ch'_1,\, ch'_2, \ldots, ch'_t) = H'(E^{(1)}||\ldots||E^{(t)}||msg)$
- if $(ch_1, ch_2, \ldots, ch_t) = (ch'_1,\, ch'_2, \ldots, ch'_t)$ returns 1, else returns 0.

Correctness. For all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and all honestly generated signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},\, msg)$ it holds that $\mathsf{Verify}(\mathsf{vk}, msg,\, \sigma) = 1$ as

$E^{(i)} = [a'_i]E_0$ is recovered by computing $[z_i]E_{ch_i}$ for $i = 1, 2, \ldots, t$ which follows from:

$$\begin{aligned}[z_i]E_{ch_i} &= [\, a'_i - \mathsf{sign}(ch_i)\, a_{|\, ch_i\, |}\,]E_{ch_i} \\ &= [\, a'_i - \mathsf{sign}(ch_i)\, a_{|\, ch_i\, |}\,][\, \mathsf{sign}(ch_i)\, a_{|\, ch_i\, |}\,]E_0 \\ &= [a'_i]E_0\end{aligned}$$

Theorem 9. *Assume the hash functions H' used are modeled as quantum random oracles. Then the signature scheme CSI-FiSh is Strong Existential Unforgeability under Chosen Message Attack (sEUF-CMA) secure.*

3 Key-Oblivious Encryption

3.1 Syntax

Definition 14. *(Key-oblivious encryption). A key-oblivious encryption (KOE) scheme is a tuple KOE = (Setup, KeyGen, KeyRand, Enc, Dec) of five polynomial-time algorithms with the following requirements:*

Setup(1^λ) $\rightarrow$ pp : This is a randomized algorithm run by a trusted authority that takes as input the security parameter λ and outputs the public parameter pp.

KeyGen(pp) $\rightarrow$ (pk, sk): A user runs this randomized algorithm on input the public parameter pp and generates a key pair (pk, sk). The public key pk is published while the secret key sk is kept secret to the user.

KeyRand(pp, pk; r) $\rightarrow$ pk$'$: Any entity can randomize pk using the public parameter pp and some randomness r to produce a new public key pk$'$ for the same secret key sk.

Enc(pp, pk, m) $\rightarrow$ ct: This randomized algorithm is executed by an encryptor who uses the public parameter pp, the public key pk of the recipient to encrypt a message m and outputs a ciphertext ct.

Dec(pp, ct, sk) $\rightarrow$ m': This is a deterministic algorithm run by the decryptor taking the secret key sk, the ciphertext ct, the public parameter pp and outputs the decrypted message m'.

Correctness. A KOE scheme is said to be correct if for all security parameter λ, all pp $\leftarrow$ Setup(1^λ), all (pk, sk) $\leftarrow$ KeyGen(pp), all pk$'$ $\leftarrow$ KeyRand(pp, pk; r), all m, it must hold that Dec(pp, Enc(pp, pk$'$, m), sk) $= m$.

3.2 Security model

We describe below the three security requirements for KOE: (i) *key randomizability* (KR) (ii) *plaintext indistinguishability under key randomization* (INDr) and (iii) *key privacy under key randomization* (KPr).
(i) *Key randomizability* (KR): This property demands that no adversary should be able to figure out how the public keys are related to each other without the secret key and randomness used. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 2

$\underline{\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}}$

1. The challenger $\mathcal{C}$ performs the following steps to generate (pp, pk, pk_b) and sends it to the adversary $\mathcal{A}$
   - $\mathsf{pp} \leftarrow \mathsf{KOE.Setup}(1^\lambda)$
   - $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KOE.KeyGen}(\mathsf{pp})$
   - $\mathsf{pk}_0 \leftarrow \mathsf{KOE.KeyRand}(\mathsf{pp}, \mathsf{pk}; r)$
   - $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KOE.KeyGen}(\mathsf{pp})$
   - $b \leftarrow \{0,1\}$
2. The adversary $\mathcal{A}$ eventually outputs a guess bit $b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{pk}_b)$ where $b' \in \{0,1\}$.
3. The challenger $\mathcal{C}$ returns 1 if $b = b'$ and 0 otherwise.

Fig. 2. The key randomizability experiment $\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}$

Definition 15. *(Key randomizability). A KOE scheme is key randomizable if the key randomizability advantage $\mathit{Adv}^{KR}_{KOE,\mathcal{A}}(\lambda) = |Pr[\mathsf{Exp}^{KR}_{KOE,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ of any PPT adversary $\mathcal{A}$ is negligible.*

(ii) *Plaintext indistinguishability under key randomization* (INDr): This security attribute necessitates that no adversary can differentiate the ciphertexts corresponding to messages of their choice even though the adversary is permitted to randomize the public key. This is formally modelled via the experiment $\mathsf{Exp}^{\mathsf{INDr}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$ given in Fig. 3 between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Definition 16. *(Plaintext indistinguishability under key randomization). A KOE scheme is plaintext indistinguishable under key randomization if the plaintext indistinguishing advantage $\mathit{Adv}^{INDr}_{KOE,\mathcal{A}}(\lambda) = |Pr[\mathsf{Exp}^{INDr}_{KOE,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ of any PPT adversary $\mathcal{A}$ is negligible.*

(iii) *Key privacy under key randomization* (KPr): This feature requires that no adversary can distinguish between ciphertexts of a particular message under adversarially randomized public keys. This is formally modelled by the experiment $\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$ in Fig 4.

$\underline{\mathsf{Exp}^{\mathsf{INDr}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda)}$

1. The challenger $\mathcal{C}$ generates (pp, pk) by performing the following steps and sends it to the adversary $\mathcal{A}$.
   - pp $\leftarrow$ KOE.Setup(1^λ)
   - (pk, sk) $\leftarrow$ KOE.KeyGen(pp)
2. The adversary $\mathcal{A}$ randomizes pk using randomness r to produce a randomized public key pk′ $\leftarrow$ KOE.KeyRand(pp, pk; r). He chooses two equal-length messages m_0, m_1 and sends (pk′, r, m_0, m_1, st) $\leftarrow \mathcal{A}$(pp, pk) to the challenger $\mathcal{C}$, where st is the state variable that stores the state information.
3. Challenger $\mathcal{C}$ performs the following steps:
   - if pk′ $\neq$ KOE.KeyRand(pp, pk; r) then return $\perp$
   - $b \leftarrow \{0,1\}$
   - ct $\leftarrow$ KOE.Enc(pp, pk′, m_b)
   - Sends ct to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ eventually outputs a guess bit $b' \leftarrow \mathcal{A}$(ct, st), where $b' \in \{0,1\}$.
5. The challenger $\mathcal{C}$ returns 1 if $b = b'$ and 0 otherwise.

Fig. 3. The plaintext indistinguishability experiment $\mathsf{Exp}^{\mathsf{INDr}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda)$

Definition 17. *(Key privacy under key randomization) A* KOE *scheme is key private under key randomization if the key privacy advantage* $\mathit{Adv}^{\mathit{KPr}}_{\mathit{KOE},\,\mathcal{A}}(\lambda) = |\mathit{Pr}[\mathsf{Exp}^{\mathit{KPr}}_{\mathit{KOE},\,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ *of any PPT adversary* $\mathcal{A}$ *is negligible.*

3.3 Key-Oblivious Encryption from Isogenies

In this section, we explain our proposed isogeny-based KOE scheme which we call it as *Commutative Supersingular Isogeny Key-Oblivious Encryption* (CSIKOE) scheme.
We will require the following notation adapted from [DFM20] where $\mathsf{Cl}(\mathcal{O})$ is the ideal class group with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ and $E \in \mathsf{Ell}_p(\mathcal{O})$, which denotes the set of $\mathbb{F}_p$-isomorphic classes of supersingular elliptic curves E, whose $\mathbb{F}_p$-endomorphism ring $\mathsf{End}_{\mathbb{F}_p}(E) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

- $[\mathfrak{a}]E$ will be replaced by $[\mathsf{a}]E$, where $[\mathfrak{a}] = [\mathfrak{l}_1^{a_1} \ldots \mathfrak{l}_n^{a_n}] \in \mathsf{Cl}(\mathcal{O})$ is determined by its exponent vector $\mathsf{a} = (a_1, \ldots, a_n)$.
- $[\mathfrak{a}][\mathfrak{b}]E$ will be replaced by $[\mathsf{a}+\mathsf{b}]E$ where $[\mathfrak{a}]$, $[\mathfrak{b}] \in \mathsf{Cl}(\mathcal{O})$ and $\mathsf{a} = (a_1, \ldots, a_n)$, $\mathsf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}^n$ represents exponent vectors of $[\mathfrak{a}]$ and $[\mathfrak{b}]$ respectively. This follows immediately from : $[\mathfrak{a}][\mathfrak{b}]E = [\mathfrak{l}_1^{a_1}\mathfrak{l}_2^{a_2} \ldots \mathfrak{l}_n^{a_n}][\mathfrak{l}_1^{b_1}\mathfrak{l}_2^{b_2} \ldots \mathfrak{l}_n^{b_n}]E = [\mathfrak{l}_1^{a_1+b_1}\mathfrak{l}_2^{a_2+b_2} \ldots \mathfrak{l}_n^{a_n+b_n}]E$.

$\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: A trusted authority runs this algorithm on input a security parameter λ and performs the following steps:

- Chooses a large prime p of the form $p = 4\,l_1 l_2 \ldots l_n - 1$, where the l_i are small distinct odd primes, picks an integer m such that $2m+1 \geq \sqrt[n]{\#\mathsf{Cl}(\mathcal{O})}$ and selects a base elliptic curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.
- As pointed out in Theorem 6, each isomorphism class of a curve with given endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ is represented by a unique Montgomery

$\underline{\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda)}$

1. The challenger $\mathcal{C}$ generates $(\mathsf{pp},\ \mathsf{pk}_0,\ \mathsf{pk}_1)$ performing the following steps and sends it to the adversary $\mathcal{A}$.
   - $\mathsf{pp} \leftarrow \mathsf{KOE.Setup}(1^\lambda)$
   - $(\mathsf{pk}_0,\ \mathsf{sk}_0) \leftarrow \mathsf{KOE.KeyGen}(\mathsf{pp})$
   - $(\mathsf{pk}_1,\ \mathsf{sk}_1) \leftarrow \mathsf{KOE.KeyGen}(\mathsf{pp})$
2. The adversary $\mathcal{A}$ randomizes pk_0 and pk_1 using randomness r_0 and r_1 respectively, to generate randomized public keys $\mathsf{pk}_0' \leftarrow \mathsf{KeyRand}(\mathsf{pp},\ \mathsf{pk}_0; r_0)$ and $\mathsf{pk}_1' \leftarrow \mathsf{KeyRand}(\mathsf{pp},\ \mathsf{pk}_1; r_1)$. He chooses a message m and sends $(\mathsf{pk}_0',\ r_0,\ \mathsf{pk}_1',\ r_1,\ m,\ st) \leftarrow \mathcal{A}(\mathsf{pp},\ \mathsf{pk}_0,\ \mathsf{pk}_1)$ to $\mathcal{C}$, where st is the variable that stores the state information.
3. Challenger $\mathcal{C}$ performs the following steps:
   - if $\mathsf{pk}_i' \neq \mathsf{KOE.KeyRand}(\mathsf{pp},\ \mathsf{pk}_i;\ r_i)$ for some $i \in \{0,1\}$ then return $\perp$
   - $b \leftarrow \{0,1\}$
   - $\mathsf{ct} \leftarrow \mathsf{KOE.Enc}(\mathsf{pp},\ \mathsf{pk}_b',\ m)$
   - $\mathcal{C}$ sends ciphertext ct to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ eventually outputs a guess bit $b' \leftarrow \mathcal{A}(\mathsf{ct},\ st)$, where $b' \in \{0,1\}$.
5. The challenger $\mathcal{C}$ returns 1 if $b = b'$ and 0 otherwise.

Fig. 4. The key privacy under key randomization experiment $\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda)$

coefficient $A_{\mathsf{mg}} \in \mathbb{F}_p$ defining the curve $E_{A_{\mathsf{mg}}} : y^2 = x^3 + A_{\mathsf{mg}}x^2 + x$. Thus, the trusted authority defines $M_C : \mathsf{Ell}_p(\mathcal{O}) \rightarrow \mathbb{F}_p$, a function that maps isomorphism classes of elliptic curve to its Montgomery coefficient.

- Samples a family of keyed hash function $\mathcal{H} := \{H_k\}_{k\in K}$ where $H_k : \mathbb{F}_p \rightarrow \{0,1\}^\lambda$ for each $k \in K$, where K is the key space and let the message space be $\mathcal{M} = \{0,1\}^\lambda$.
- Sets the public parameter $\mathsf{pp} = (p, l_1, l_2, \ldots, l_n, m, E_0, M_C, \mathcal{H} := \{H_k\}_{k\in K})$.

$\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk},\ \mathsf{sk})$: This is a randomized algorithm run by a user on input the public parameter $\mathsf{pp} = (p,\ l_1,\ l_2,\ \ldots,\ l_n,\ m,\ E_0,\ M_C,\ \mathcal{H} := \{H_k\}_{k\in K})$ to generate his public key pk and secret key sk. The user executes the following steps:

- Samples randomly two n-tuple integer vectors $\mathsf{a} = (a_1, a_2, \ldots, a_n)$, $\mathsf{r} = (r_1, r_2 \ldots, r_n)$ where $a_i,\ r_i \in [-m,\ m]$ for $i = 1, 2, \ldots, n$. These integer vectors define the ideal classes $[\mathfrak{a}] = [\mathfrak{l}_1^{a_1}\mathfrak{l}_2^{a_2}\ldots\mathfrak{l}_n^{a_n}]$ and $[\mathfrak{r}] = [\mathfrak{l}_1^{r_1}\mathfrak{l}_2^{r_2}\ldots\mathfrak{l}_n^{r_n}]$ $\in \mathsf{Cl}(\mathcal{O})$ respectively, where $\mathfrak{l}_i = < l_i, \pi - 1 >$. Here the notation $<,>$ denotes the ideal generated by multiplication by l_i map and $\pi - 1$ where π is the Frobenius endomorphism.
- Computes the elliptic curves $E_1 = [\mathsf{a}]E_0$, $E_2 = [\mathsf{r}]E_1 = [\mathsf{r} + \mathsf{a}]E_0$ and returns the public key $\mathsf{pk} = (E_1,\ E_2)$ and keeps $\mathsf{sk} = \mathsf{r}$ secret.

$\mathsf{KeyRand}(\mathsf{pp},\ \mathsf{pk};\ \mathsf{r}') \rightarrow \mathsf{pk}'$: This randomized algorithm takes input the public parameter pp, public key $\mathsf{pk} = (E_1, E_2)$ and randomize it to obtain pk'. The steps involved are as follows:

– Samples randomly an n-tuple integer vector $\mathsf{r}' = (r'_1, r'_2, \ldots, r'_n)$ where $r'_i \in [-m,\ m]$ for $i = 1, 2, \ldots, n$. This integer vector r' defines the ideal class $[\mathfrak{r}']$ $= [\mathfrak{l}_1^{r'_1} \mathfrak{l}_2^{r'_2} \ldots \mathfrak{l}_n^{r'_n}] \in \mathsf{Cl}(\mathcal{O})$.

– Computes the curves $E'_1 = [\mathsf{r}']E_1,\ E'_2 = [\mathsf{r}']E_2$ and outputs the randomized public key $\mathsf{pk}' = (E'_1,\ E'_2)$.

$\mathsf{Enc}(\mathsf{pp},\ \mathsf{pk},\ m) \to \mathsf{ct}$: An encryptor takes input the public parameter pp, the public key $\mathsf{pk} = (E_1, E_2)$, a message m and performs the following steps:

– Samples randomly an n-tuple vector $\mathsf{c} = (c_1, c_2 \ldots, c_n)$ of integers, where $c_i \in [-m, m]$ for $i = 1, 2, \ldots, n$, which defines the ideal class $[\mathfrak{c}] = [\mathfrak{l}_1^{c_1} \mathfrak{l}_2^{c_2} \ldots \mathfrak{l}_n^{c_n}]$ $\in \mathsf{Cl}(\mathcal{O})$.

– Computes $\mathsf{ct}_1 = [\mathsf{c}]E_1,\ \mathsf{ct}_2 = H_k(M_C([\mathsf{c}]E_2)) \oplus m$ and returns the ciphertext $\mathsf{ct} = (\mathsf{ct}_1,\ \mathsf{ct}_2)$.

$\mathsf{Dec}(\mathsf{pp},\ \mathsf{ct},\ \mathsf{sk}) \to m$: This a deterministic algorithm run by a decryptor that takes input the public parameter pp, the secret key $\mathsf{sk} = \mathsf{r}$ and the ciphertext $\mathsf{ct} = (\mathsf{ct}_1,\ \mathsf{ct}_2)$ where $\mathsf{ct}_1 = [\mathsf{c}]E_1,\ \mathsf{ct}_2 = H_k(M_C([\mathsf{c}]E_2)) \oplus m$. The decryptor retrieves the plaintext m by computing $\mathsf{ct}_2 \oplus H_k(M_C([\mathsf{r}]\mathsf{ct}_1))$.

Correctness. The ciphertext of the CSIKOE protocol under randomized public key $\mathsf{pk}' = (E'_1,\ E'_2)$ is given by : $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) = ([\mathsf{c}]E'_1,\ H_k(M_C([\mathsf{c}]E'_2)) \oplus m)$. Consequently we have,

$$\begin{aligned}\mathsf{ct}_2 \oplus H_k(M_C([\mathsf{r}]\mathsf{ct}_1)) &= H_k(M_C([\mathsf{c}]E'_2)) \oplus m \oplus H_k(M_C([\mathsf{r}]\mathsf{ct}_1)) \\ &= H_k(M_C([\mathsf{c}][\mathsf{r}'][\mathsf{r}+\mathsf{a}]E_0)) \oplus m \oplus H_k(M_C([\mathsf{r}][\mathsf{c}][\mathsf{r}']E_1)) \\ &= m.\end{aligned}$$

Remark 2. Castryck et al. [CSV20] points out that the CSSDDH problem is easy if we work with supersingular elliptic over $\mathbb{F}_p$ with $p \equiv 1 \pmod 4$. Thus such a choice of curve is not recommended for our construction. It is noteworthy that CSIDH is secure as it relies on supersingular elliptic curves over $\mathbb{F}_p$ with $p \equiv 3 \pmod 4$. Consequently, our CSIKOE construction is secure as for our setting the CSSDDH assumption is conjectured to be hard. (See Remark 1)

Parameter setting for security. The system parameters must be set in such a way that no polynomial-time adversary can guess the private key with non-negligible probability. Note that the private key is an n-tuple vector of integers with each co-ordinates chosen randomly from $[-m,\ m]$. Therefore the private key space is $(2m+1)^n$ and $(2m+1)^n \geq 2^{3\lambda} \Rightarrow n \log(2m+1) \geq 3\lambda$ needs to be satisfied to provide a secure key space, with the goal that no polynomial-time attacker can guess the private key [DFG19]. Considering the best-known threats, three sets of parameters were recommended for CSIDH under three NIST security levels - CSIDH-512, CSIDH-1024 and CSIDH-1792. The parameters of CSIDH-512 were fully specified in practice ($n = 74,\ m = 5,\ l_{73} = 373,\ l_{74} = 587$) corresponding to the NIST level 1 and it could achieve 127-bit classical and 64-bit quantum security.

Efficiency. We now analyse the efficiency of our CSIKOE scheme in terms of security parameter λ. The size of public key pk is of order $O(\log p) = O(\lambda)$. The secret key sk has $n \log(2m+1)$ bits. Since $n \log(2m+1) \geq 3\lambda$, thus the size of the secret key sk is of order $O(\lambda)$. The storage, communication cost and computation cost have been summed up in the table below.

Storage		Communication cost	Computation cost	
$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	Encryption	Decryption
$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	2 group actions 1 XOR operation	1 group action 1 XOR operation

3.4 Security Analysis

Theorem 10. *Under the CSSDDH assumption as defined in Definition 7 of section 2, the isogeny based CSIKOE scheme presented in section 3.3 satisfies key randomizability (KR) as per Definition 15.*

Proof. Let us assume that there exists a PPT adversary $\mathcal{A}$ and a non-negligible function $\mu(\cdot)$ such that $\Pr[\,\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}(\lambda) = 1\,] \geqslant \frac{1}{2} + \mu(\lambda)$, where $\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$ is defined in Fig. 2 of section 3. We will prove that we can design a PPT distinguisher $\mathcal{D}$ which can solve any CSSDDH instance, i.e., distinguishes between $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0\,)$ and $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_0 = [\mathsf{z}]E_0\,)$ with non-negligible probability where $E_0 \in \mathsf{Ell}_p(\mathcal{O})$ and $\mathsf{x} = (x_1,\, x_2, \ldots, x_n)$, $\mathsf{y} = (y_1,\, y_2, \ldots, y_n)$, $\mathsf{z} = (z_1,\, z_2, \ldots, z_n)$ are integer vectors such that $x_i,\, y_i,\, z_i \in [-m,\, m]$ for $i = 1, 2, \ldots, n$. Given a CSSDDH challenge $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_b\,)$ where $b \in \{0,1\}$, the reduction is straight forward and proceeds as described in Fig. 5.

1. The Distinguisher $\mathcal{D}$ uses the CSSDDH instance $(E_0,\, X,\, Y,\, Z_b)$ where $b \in \{0,1\}$, from the CSSDDH challenger, generates $(\mathsf{pk},\, \mathsf{pk}_b)$ and sends it to $\mathcal{A}$.
   - Sets $\mathsf{pp} = (p,\, E_0, l_1,\, l_2,\, \ldots,\, l_n,\, m,\, M_C,\, \mathcal{H} := \{H_k\}_{k \in K})$ where $p,\, E_0$ are extracted from the CSSDDH instance, $l_i's$ are odd prime, M_C is a function from $\mathsf{Ell}_p(\mathcal{O})$ to $\mathbb{F}_p$ and $\mathcal{H}$ is a family of keyed hash function.
   - Samples an integer vector $\mathsf{r} = (r_1,\, r_2, \ldots, r_n)$ such that $r_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.
   - Sets $\mathsf{pk} = ([\mathsf{r}]E_0,\, [\mathsf{r}]X)$, $\mathsf{pk}_b = ([\mathsf{r}]Y,\, [\mathsf{r}]Z_b))$.
2. The adversary $\mathcal{A}$ eventually outputs a bit $b \leftarrow \mathcal{A}(\mathsf{pk},\, \mathsf{pk}_b)$, where $b \in \{0,1\}$.
3. The distinguisher returns the bit b to the CSSDDH challenger.

Fig. 5. Distinguisher $\mathcal{D}$ for the KR security of CSIKOE

For the instance when $b = 1$, i.e., $(X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0)$ is a CSSDDH triple, the view of the adversary $\mathcal{A}$ is identical to experiment $\mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$. As adversary receives original public key $\mathsf{pk} = ([\mathsf{r}]E_0,\, [\mathsf{r}]X) =$

$([\mathsf{r}]E_0,\ [\mathsf{x}{+}\mathsf{r}]E_0)$ and the subsequent public key $\mathsf{pk}_1 = ([\mathsf{r}]Y,\ [\mathsf{r}]Z_1)) = ([\mathsf{y}][\mathsf{r}]E_0,\ [\mathsf{y}][\mathsf{x}{+}\mathsf{r}]E_0)$, a re-randomization of the original key pk using y. On the other hand, when $b = 0$, i.e., $(X = [\mathsf{x}]E_0,\ Y = [\mathsf{y}]E_0,\ Z_0 = [\mathsf{z}]E_0)$, the second key $\mathsf{pk}_0 = ([\mathsf{r}]Y,\ [\mathsf{r}]Z_0)) = ([\mathsf{y}][\mathsf{r}]E_0,\ [\mathsf{z}{+}\mathsf{r}]E_0)$ is a complete randomized key. Thus, if $\mathcal{A}$ correctly distinguishes between a real or random key with a non-negligible advantage, the distinguisher $\mathcal{D}$ breaks CSSDDH with the same non-negligible advantage as that of $\mathcal{A}$. More formally, the probability of $\mathcal{D}$ winning in the distinguishability game $= \Pr[\ \mathsf{Exp}^{\mathsf{KR}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda) = 1\ ] \geqslant \frac{1}{2} + \mu(\lambda)$. This completes the proof. □

Theorem 11. *The isogeny-based* CSIKOE *scheme presented in Section 3.3 satisfies plaintext indistinguishable under key randomization (*INDr*) as per Definition 16 under* CSSDDH *assumption as defined in Definition 7 and the assumption that the family of hash functions $\mathcal{H} := \{H_k\}_{k\in K}$ is "entropy smoothing" as defined in Definition 8.*

Proof. We prove the plaintext indistinguishability under key randomization (INDr) of our CSIKOE scheme using the following sequence of games G_0, G_1, G_2, under CSSDDH assumption and the presumption that $\mathcal{H}$ is entropy smoothing.
Game G_0. We start with game G_0 which is the true INDr experiment $\mathsf{Exp}^{\mathsf{INDr}}_{\mathsf{KOE},\,\mathcal{A}}(\lambda)$ and is explicitly described in Fig. 6.

1. The challenger $\mathcal{C}$ begins the experiment by computing (pp, pk) and sends it to the adversary $\mathcal{A}$.
   - $\mathsf{pp} \leftarrow \mathsf{KOE.Setup}(1^\lambda)$ where $\mathsf{pp} = (p,\ E_0, l_1,\ l_2,\ \ldots,\ l_n,\ m,\ M_C,\ \mathcal{H} := \{H_k\}_{k\in K})$ where p is a prime of the form $4l_1l_2\ldots l_n$ where $l_i's$ are odd prime, E_0 is the basic elliptic curve, M_C is function from $\mathsf{Ell}_p(\mathcal{O})$ to $\mathbb{F}_p$ and $\mathcal{H}$ is a family of keyed hash function.
   - Samples two integer vector $\mathsf{r} = (r_1,\ r_2, \ldots, r_n)$, $\mathsf{a} = (a_1,\ a_2, \ldots, a_n)$ such that $r_i,\ a_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.
   - $(\mathsf{pk} = (E_1 = [\mathsf{a}]E_0,\ E_2 = [\mathsf{r}{+}\mathsf{a}]E_0),\ \mathsf{sk} = \mathsf{r}) \leftarrow \mathsf{KOE.KeyGen}(\mathsf{pp})$
2. The adversary $\mathcal{A}$ randomizes $\mathsf{pk} = (E_1,\ E_2)$ using randomness r' to produce a randomized public key $\mathsf{pk}' \leftarrow \mathsf{KOE.KeyRand}(\mathsf{pp},\ \mathsf{pk};\ \mathsf{r}')$. He chooses two equal-length messages $m_0,\ m_1$ and sends $(\mathsf{pk}',\ \mathsf{r}',\ m_0,\ m_1,\ st) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk})$ to the challenger $\mathcal{C}$, where $\mathsf{pk}' = (\ E_1' = [\mathsf{r}']E_1,\ E_2' = [\mathsf{r}']E_2]\ )$ and st is the state variable that stores the state information.
3. Challenger $\mathcal{C}$ performs the following steps:
   - if $\mathsf{pk}' \neq \mathsf{KOE.KeyRand}(\mathsf{pp},\ \mathsf{pk};\ \mathsf{r}')$ then return $\perp$
   - $b \leftarrow \{0, 1\}$
   - Samples an integer vector $\mathsf{c} = (c_1,\ c_2, \ldots, c_n)$ such that $c_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.
     - For game G_0, $\mathsf{ct} \leftarrow \mathsf{KOE.Enc}(\mathsf{pp},\ \mathsf{pk}', m_b)$, where $\mathsf{ct} = (\mathsf{ct}_1 = [\mathsf{c}]E_1',\ \mathsf{ct}_2 = H_k(\ M_C(\ [\mathsf{c}]E_2'\ )\ ) \oplus m_b)$
     - For game G_1, $\mathsf{ct} = (\mathsf{ct}_1 = [\mathsf{c}]E_1',\ \mathsf{ct}_2 = H_k(\ M_C(\ [\mathsf{z}]E_2'\ )\ ) \oplus m_b)$ where $\mathsf{z} = (z_1,\ z_2, \ldots, z_n)$ is an integer vector such that $z_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.
     - For game G_2, $\mathsf{ct} = (\mathsf{ct}_1 = [\mathsf{c}]E_1',\ \mathsf{ct}_2 = h \oplus m_b)$ where $h \leftarrow \{0, 1\}^\lambda$.
   - Challenger $\mathcal{C}$ sends ciphertext ct to the adversary $\mathcal{A}$.
4. The adversary $\mathcal{A}$ eventually outputs a guess bit $b' \leftarrow \mathcal{A}(\mathsf{ct},\ st)$, where $b' \in \{0, 1\}$.
5. The challenger $\mathcal{C}$ returns 1 if $b = b'$ and 0 otherwise.

Fig. 6. Game G_i for $i = 0,\ 1,\ 2$ in the proof of Theorem 11

Game G_1. It is the same as game G_0, but with a small tweak. In this game, the challenger sets the targeted ciphertext $\mathsf{ct} = (\mathsf{ct}_1 = [\mathsf{c}]E_1',\ \mathsf{ct}_2 = H_k(\ M_C(\ [\mathsf{z}]E_2'\ )\ )\oplus$

m_b) where $\mathsf{z} = (z_1, z_2, \ldots, z_n)$ is an integer vector sampled randomly such that $z_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.

For Game G_i for $i = 0, 1, 2$, let T_i be the event associated with $b = b'$. We assume that $\mathcal{A}$ submits two messages m_0 and m_1 of equal-length in each game. We first prove the following claim.
Claim: For the event T_0 in game G_0 and event T_1 in game G_1, we have $|\,\mathsf{Pr}[T_0] - \mathsf{Pr}[T_1]\,| = \epsilon_{\mathsf{cssddh}}$ where $\epsilon_{\mathsf{cssddh}}$ is the CSSDDH-advantage of any PPT adversary, which is negligible.
Proof of claim. To prove that $|\,\mathsf{Pr}[T_0] - \mathsf{Pr}[T_1]\,|$ is negligible, one argues that there exists a distinguishing algorithm $\mathcal{D}$ that interpolates between game G_0 and game G_1, so that when given $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0\,)$ as input, $\mathcal{D}$ outputs 1 with probability $\mathsf{Pr}[T_0]$ and when given $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_0 = [\mathsf{z}]E_0\,)$ as input, $\mathcal{D}$ outputs 1 with probability $\mathsf{Pr}[T_1]$. The CSSDDH indistinguishability assumption then implies that $|\,\mathsf{Pr}[T_0] - \mathsf{Pr}[T_1]\,|$ is negligible. Our distinguisher $\mathcal{D}$ is precisely described in Fig. 7.

1. The Distinguisher $\mathcal{D}$ uses the CSSDDH instance $(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_\delta\,)$ where $\delta \in \{0, 1\}$ from the CSSDDH challenger, generates (pp, pk) and sends it to the adversary $\mathcal{A}$
   - Sets $\mathsf{pp} = (p,\, E_0, l_1,\, l_2,\, \ldots,\, l_n,\, m,\, M_C,\, \mathcal{H} := \{H_k\}_{k\in K})$ where p, E_0 are extracted from the CSSDDH instance, M_C is function from $\mathsf{Ell}_p(\mathcal{O})$ to $\mathbb{F}_p$ and $\mathcal{H}$ is a family of keyed hash function.
   - Samples an integer vector $\mathsf{a} = (a_1,\, a_2, \ldots, a_n)$ such that $a_i \in [-m, m]$ for $i = 1, 2, \ldots, n$.
   - $\mathsf{pk} \leftarrow ([\mathsf{a}]E_0,\, [\mathsf{a}]X)$.
2. The adversary $\mathcal{A}$ randomizes pk using randomness r' to produce a randomized public key pk' $\leftarrow$ KOE.KeyRand(pp, pk; r'). He chooses two equal-length messages m_0, m_1 and sends (pk', r', m_0, m_1, st) $\leftarrow \mathcal{A}$(pp, pk) to the distinguisher $\mathcal{D}$, where $\mathsf{pk}' = (\,[\mathsf{r}'][\mathsf{a}]E_0,\, [\mathsf{r}'][\mathsf{a}{+}\mathsf{x}]E_0]\,)$ and st is the state variable that stores the state information.
3. The distinguisher $\mathcal{D}$ performs the following steps:
   - if $\mathsf{pk}' \neq (\,[\mathsf{r}'][\mathsf{a}]E_0,\, [\mathsf{r}'][\mathsf{a}]X)$ then return $\perp$
   - $b \leftarrow \{0, 1\}$
   - $\mathsf{ct} \leftarrow ([\,\mathsf{a} + \mathsf{r}']Y,\; H_k(M_C([\mathsf{a} + \mathsf{r}']Z_\delta)) \oplus m_b)$; sends ciphertext ct to the adversary $\mathcal{A}$.
4. Adversary $\mathcal{A}$ taking the ciphertext ct eventually outputs a bit $b' \leftarrow \mathcal{A}(\mathsf{ct},\, st)$, where $b' \in \{0, 1\}$.
5. The distinguisher $\mathcal{D}$ outputs 1 if $b = b'$ or else outputs 0.

Fig. 7. Distinguisher $\mathcal{D}$ for the INDr security of CSIKOE

If the input to $\mathcal{D}$ is of the form $(\,E_0,\, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0\,)$, then computation proceeds just as in game G_0, and therefore
$\mathsf{Pr}[\,\mathsf{x},\,\mathsf{y} \leftarrow [-m,\, m]^n \mid \mathcal{D}(\,E_0,\, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0\,) = 1] = \mathsf{Pr}[T_0]$.
On the other hand, if the input to $\mathcal{D}$ is of the form $(\,E_0,\, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0,\, Z_0 = [\mathsf{z}]E_0\,)$, then computation proceeds just as in game G_1, and therefore $\mathsf{Pr}[\,\mathsf{x},\,\mathsf{y} \leftarrow [-m,\, m]^n \mid \mathcal{D}(\,E_0,\, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0,\, Z_0 = [\mathsf{z}]E_0\,) = 1] = \mathsf{Pr}[T_1]$.
Thus we have,

$$
\begin{aligned}
&\mathsf{Adv}_{\mathcal{D}}^{\mathsf{CSSDDH}}(\lambda) \\
&= |\,\mathsf{Pr}[\mathsf{x},\,\mathsf{y} \leftarrow [-m,\, m]^n \,|\, \mathcal{D}(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_1 = [\mathsf{x}{+}\mathsf{y}]E_0\,) = 1] \\
&\qquad - \mathsf{Pr}[\mathsf{x},\,\mathsf{y} \leftarrow [-m,\, m]^n \,|\, \mathcal{D}(\,E_0,\, X = [\mathsf{x}]E_0,\, Y = [\mathsf{y}]E_0,\, Z_0 = [\mathsf{z}]E_0\,) = 1]\,|
\end{aligned}
$$

$= |\,\mathsf{Pr}[T_0] - \mathsf{Pr}[T_1]\,|$
From this, it follows that the CSSDDH-advantage of $\mathcal{D}$ is equal to $|\,\mathsf{Pr}[T_0] - \mathsf{Pr}[T_1]\,|$, which completes the proof of the Claim.
Game G_2. Game G_2 is identical to game G_1, except that the challenger sets $\mathsf{ct} = (\mathsf{ct}_1 = [\mathsf{c}]E_1', \mathsf{ct}_2 = h \oplus m_b)$ by choosing $h \in \{0,1\}^\lambda$ uniformly at random. Then from the entropy smoothing assumption of the family of hash functions $\mathcal{H}$ as defined in Definition 8, we have $|\,\mathsf{Pr}[T_1]-\mathsf{Pr}[T_2]\,| = \epsilon_{\mathsf{es}}$, where ϵ_{es} is the entropy smoothing advantage of any PPT algorithm, which is negligible. Also, note that as h behaves like a one-time pad in game G_2. Thus, $\mathsf{Pr}[\,T_2\,] = \frac{1}{2}$. Hence, we get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{INDr}}_{\mathsf{KOE},\mathcal{A}}(\lambda) &= |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{INDr}}_{\mathsf{KOE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \\ &= |\mathsf{Pr}[\,T_0\,] - \mathsf{Pr}[\,T_2\,]| \\ &\leq |\mathsf{Pr}[\,T_0\,] - \mathsf{Pr}[\,T_1\,]| + |\mathsf{Pr}[\,T_1\,] - \mathsf{Pr}[\,T_2\,]| \\ &= \epsilon_{\mathsf{cssddh}} + \epsilon_{\mathsf{es}}\end{aligned}$$

which is negligible since both $\epsilon_{\mathsf{cssddh}}$ and ϵ_{es} are negligible. This completes the proof. ☐

Theorem 12. *Under the CSSDDH assumption as defined in Definition 7 of section 2, the isogeny based CSIKOE scheme presented in section 3.3 satisfies key private under key randomization (KPr) as per Definition 17.*

Proof. On the contrary, let us assume that there exists a PPT adversary $\mathcal{A}$ and a non-negligible function $\mu(\cdot)$ such that $\mathsf{Pr}[\,\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\mathcal{A}}(\lambda) = 1\,] \geqslant \frac{1}{2} + \mu(\lambda)$. Now we shall prove that we can design a PPT distinguisher $\mathcal{D}$ which distinguishes between $(E_0, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0, Z_1 = (\,[\mathsf{x}+\mathsf{y}]E_0)$ and $(E_0, X = [\mathsf{x}]E_0, Y = [\mathsf{y}]E_0, Z_0 = (\,[\mathsf{z}]E_0)$ with non negligible probability where $E_0 \in \mathsf{Ell}_p(\mathcal{O})$ and $\mathsf{x} = (x_1, x_2, \ldots, x_n)$, $\mathsf{y} = (y_1, y_2, \ldots, y_n)$, $\mathsf{z} = (z_1, z_2, \ldots, z_n)$ are integer vectors such that $x_i, y_i, z_i \in [-m, m]$ for $i = 1, 2, \ldots, n$. Given a CSSDDH challenge $(\,E_0, X, Y, Z_\delta\,)$, where $\delta \in \{0,1\}$ the reduction is given in the Fig. 8.

Observe that if X_0, Y_0, S_0 is a CSSDDH triple, then so is X_1, Y_1, S_1. Moreover, the two triples are identically distributed and generates proper distributions of keys in CSIKOE. For the instance when $\delta = 1$, $(\,X_b = [\mathsf{x}_b]E_0, Y_b = [\mathsf{y}_b]E_0, S_b = [\mathsf{x}_b + \mathsf{y}_b]E_0\,)$ is a CSSDDH triple, the view of the adversary $\mathcal{A}$ is identical to experiment $\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$, where $\mathsf{x}_0 = \mathsf{x}$, $\mathsf{x}_1 = \mathsf{x} + \boldsymbol{\alpha}$, $\mathsf{y}_0 = \mathsf{y}$, $\mathsf{y}_1 = \mathsf{y} + \boldsymbol{\beta}$. Indeed, $[\mathsf{r}_b + \mathsf{v}_b]S_b = [\mathsf{r}_b + \mathsf{v}_b][\mathsf{x}_b + \mathsf{y}_b]E_0 = [\mathsf{y}_b]([\mathsf{r}_b + \mathsf{v}_b]X_b) = [\mathsf{y}_b]\mathsf{ct}_1$. On the other hand, when $\delta = 0$, S_b is a random element, then the challenge ciphertext provided to $\mathcal{A}$ contains no information. Hence $\mathcal{A}$'s advantage at guessing the bit is negligible. Thus, if $\mathcal{A}$ has a non-negligible advantage in experiment $\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\mathcal{A}}(\lambda)$, $\mathcal{D}$ breaks CSSDDH with the same non-negligible advantage as that of $\mathcal{A}$. Thus, the probability of $\mathcal{D}$ winning in the distinguishability game $= \Pr[\,\mathsf{Exp}^{\mathsf{KPr}}_{\mathsf{KOE},\mathcal{A}}(\lambda) = 1\,] \geqslant \frac{1}{2} + \mu(\lambda)$. ☐

1. The Distinguisher $\mathcal{D}$ uses the CSSDDH instance (E_0, $X = [\mathsf{x}]E_0$, $Y = [\mathsf{y}]E_0$, Z_δ), where $\delta \in \{0,1\}$ from the CSSDDH challenger and performs the following steps to generate (pp, pk_0, pk_1) and sends it to the adversary $\mathcal{A}$.
   - Sets $\mathsf{pp} = (p,\ E_0, l_1,\ l_2,\ \ldots,\ l_n,\ m,\ M_C,\ \mathcal{H} := \{H_k\}_{k\in K})$ where p, E_0 are extracted from the CSSDDH instance, $l_i's$ are odd prime, M_C is function from $\mathsf{Ell}_p(\mathcal{O})$ to $\mathbb{F}_p$ and $\mathcal{H}$ is a family of keyed hash function.
   - Samples integer vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, r_0, r_1 such that co-ordinate of each of these vectors belong to $[-m,\ m]$.
   - $X_0 = X$; $Y_0 = Y$; $S_0 = Z_\delta$
   - $X_1 = [\boldsymbol{\alpha}]X$; $Y_1 = [\boldsymbol{\beta}]Y$; $S_1 = [\boldsymbol{\alpha} + \boldsymbol{\beta}]Z_\delta$
   - $\mathsf{pk}_0 = (\,[\mathsf{r}_0]E_0,\ [\mathsf{r}_0]Y_0\,)$
   - $\mathsf{pk}_1 = (\,[\mathsf{r}_1]E_0,\ [\mathsf{r}_1]Y_1,\ )$
2. The adversary $\mathcal{A}$ randomizes the public key pk_0, pk_1 using randomness v_0, v_1 respectively and generates $\mathsf{pk}_0' = ([\mathsf{v}_0 + \mathsf{r}_0]E_0,\ [\mathsf{v}_0 + \mathsf{r}_0]Y_0)$, $\mathsf{pk}_1' = ([\mathsf{v}_1 + \mathsf{r}_1]E_0,\ [\mathsf{v}_1 + \mathsf{r}_1]Y_1)$ and sends $(\mathsf{pk}_0',\ \mathsf{pk}_1',\ \mathsf{v}_0,\ \mathsf{v}_1,\ m,\ st) \leftarrow \mathcal{A}(\mathsf{pp},\ \mathsf{pk}_0,\ \mathsf{pk}_1)$ to the distinguisher $\mathcal{D}$, where m is the message and st is the state variable that stores state information.
3. The distinguisher $\mathcal{D}$ performs the following steps:
   - $\mathsf{if}\ \mathsf{pk}_0' \neq ([\mathsf{v}_0][\mathsf{r}_0]E_0,\ [\mathsf{v}_0][\mathsf{r}_0]Y_0) \lor \mathsf{pk}_1' \neq ([\mathsf{v}_1][\mathsf{r}_1]E_0,\ [\mathsf{v}_1][\mathsf{r}_1]Y_1)\ \mathsf{return}\ \perp$
   - $b \leftarrow \{0,1\}$
   - Computes ciphertext $\mathsf{ct} = (\mathsf{ct}_1,\ \mathsf{ct}_2)$ where $\mathsf{ct}_1 = [\mathsf{r}_b + \mathsf{v}_b]X_b$, $\mathsf{ct}_2 = H_k(M_C([\mathsf{r}_b + \mathsf{v}_b]S_b\,))\oplus m$ and sends it to the adversary $\mathcal{A}$.
4. Adversary $\mathcal{A}$ taking the ciphertext ct outputs a bit $b' \leftarrow \mathcal{A}(\mathsf{ct},\ st)$, where $b' \in \{0,1\}$
5. The distinguisher $\mathcal{D}$ outputs 1 if $b = b'$ or else outputs 0.

Fig. 8. Distinguisher $\mathcal{D}$ for the KPr security of CSIKOE

4 Instantiation of CSIKOE from CSIDH-512

We now show an instantiation of our CSIKOE scheme based on the $\mathsf{CSIDH\text{-}512}$ parameter set and name it as $\mathsf{CSIKOE\text{-}512}$. The structure of the class group $\mathsf{Cl}(\mathcal{O})$ where $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ is computed by Beullens et al. [BKV19]. They have shown that $\mathsf{Cl}(\mathcal{O})$ is a cyclic group and $\mathfrak{g} = <3, \pi - 1>$ is a generator of this class group. The class number of this ideal class group is given by N, where

$$N = \#\mathsf{Cl}(\mathcal{O}) = 37\ \times\ 1407181\ \times\ 51593604295295867744293584889 \\ \times 31599414504681995853008278745587832204909.$$

Thus for simplicity we can consider class group $\mathsf{Cl}(\mathcal{O})$ to be $\mathbb{Z}_N$.
We shall use the following notations for the sake of simplicity.

- $[\mathfrak{a}]E$ will be replaced by $[a]E$ for any element $[\mathfrak{a}] \in \mathsf{Cl}(\mathcal{O})$ which can be written as $[\mathfrak{g}^a]$ for some $a \in \mathbb{Z}_N$.
- $[\mathfrak{a}][\mathfrak{b}]E$ will be replaced by $[a + b]E$ where $[\mathfrak{a}]$, $[\mathfrak{b}] \in \mathsf{Cl}(\mathcal{O})$ and $[\mathfrak{a}]E = [\mathfrak{g}^a]E$, $[\mathfrak{b}]E = [\mathfrak{g}^b]E$ for some $a,\ b \in \mathbb{Z}_N$. This follows immediately from : $[\mathfrak{a}][\mathfrak{b}]E = [\mathfrak{g}^a][\mathfrak{g}^b]E = [\mathfrak{g}^{a+b}]E$.

$\mathsf{Setup}(1^\lambda) \rightarrow\ \mathsf{pp}$: A trusted authority chooses a large prime p of the form $p = 4\,l_1 l_2 \ldots l_n - 1$, where l_i's are small distinct odd primes with $n = 74$, $l_1 = 3$, $l_{73} = 373$, and $l_{74} = 587$ and selects the base elliptic curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. He sets the generator of the ideal class group $\mathcal{G} = \mathsf{Cl}(\mathcal{O})$ to be $\mathfrak{g} = <3, \pi - 1>$ with class number N and samples a family of keyed hash function $\mathcal{H} := \{H_k\}_{k\in K}$, where $H_k : \mathbb{F}_p \rightarrow \{0,1\}^\lambda$ for each $k \in K$.

Finally he sets the message space $\mathcal{M} = \{0,1\}^\lambda$ and the public parameter $\mathsf{pp} = (p, \mathfrak{g}, N, E_0, M_C, \mathcal{H})$.

$\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$: This is a randomized algorithm run by a user to generate his corresponding pair of public and secret keys. The user samples two elements $[\mathfrak{a}] = [\mathfrak{g}^a]$ and $[\mathfrak{r}] = [\mathfrak{g}^r] \in \mathcal{G}(\cong \mathbb{Z}_N)$ for some a, r in $\mathbb{Z}_N$. Computes the elliptic curves $E_1 = [a]E_0$, $E_2 = [r]E_1 = [r+a]E_0$ and returns the public key $\mathsf{pk} = (E_1, E_2)$ and keeps $\mathsf{sk} = r$ secret.

$\mathsf{KeyRand}(\mathsf{pp}, \mathsf{pk}; r') \rightarrow \mathsf{pk}'$: This is a randomized algorithm run by any entity taking input a public key pk and randomize it to obtain pk'. For this he samples $[\mathfrak{r}'] = [\mathfrak{g}^{r'}] \in \mathcal{G}(\cong \mathbb{Z}_N)$ for some r' in $\mathbb{Z}_N$. Computes the elliptic curves $E_1' = [r']E_1$, $E_2' = [r']E_2$ and outputs the randomized public key $\mathsf{pk}' = (E_1', E_2')$.

$\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m) \rightarrow \mathsf{ct}$: The encryptor samples $[\mathfrak{c}] = [\mathfrak{g}^c] \in \mathcal{G}(\cong \mathbb{Z}_N)$ for some c in $\mathbb{Z}_N$. Computes $\mathsf{ct}_1 = [c]E_1$ and $\mathsf{ct}_2 = H_k(M_C([c]E_2)) \oplus m$ using the input public key $\mathsf{pk} = (E_1, E_2)$ and returns the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, which is the encryption of the message m.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{ct}, \mathsf{sk}) \rightarrow m$: Given the secret key $\mathsf{sk} = r$ and the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ where $\mathsf{ct}_1 = [c]E_1$ and $\mathsf{ct}_2 = H_k(M_C([c]E_2)) \oplus m$, the decryptor returns the message $m = \mathsf{ct}_2 \oplus H_k(M_C([r]\mathsf{ct}_1))$.

Correctness. The correctness of CSIKOE-512 is similar to the correctness of our CSIKOE scheme.

5 Accountable Tracing Signature

5.1 Syntax

Definition 18. *An accountable tracing signature (ATS) scheme is a tuple ATS = (Setup, GrKeyGen, UsKeyGen, Enroll, Sign, Verify, Open, Judge, Account) of nine polynomial-time algorithms with the following requirements:*

Setup$(1^\lambda) \rightarrow$ gp: This is a randomized algorithm run by a trusted authority that takes as input the security parameter λ and outputs the group parameter gp.

GrKeyGen(gp) $\rightarrow$ (gpk, gsk): The GM runs this randomized algorithm on input the group parameter gp and generates the group public key gpk which includes gp and group secret key gsk = (isk, opk) where isk is the issue key and opk is the opening key.

UsKeyGen(gp) $\rightarrow$ (upk, usk): This is a randomized algorithm run by a user that takes input the group parameter gp and generates its user public key upk and user secret key usk. The user public key upk is published while the user secret key usk is kept secret to the user.

Enroll(gp, gpk, isk, upk, tr) $\rightarrow$ (cert, w^{escrw}): The GM runs this randomized algorithm taking inputs the group parameter gp, the group public key gpk, a user public key upk, issue key isk and a trace bit tr $\in \{0,1\}$. For tr $= 0$, the *anonymity of the user is preserved whereas for* $\mathsf{tr} = 1$*, the user is traceable. Based on the choice of bit* tr*, the* GM *produces a certificate* cert *including* upk *and witness* w^{escrw}*. The* GM *sends the certificate* cert *to the user and keeps the witness* w^{escrw} *secret to himself.*

$\mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, \mathsf{usk}, msg) \rightarrow \sigma$: *This randomized algorithm is executed by a user that takes inputs the group parameter* gp*, the group public key* gpk*, a user secret key* usk*, user certificate* cert *and generates a signature* σ *on the message* msg*.*

$\mathsf{Verify}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma) \rightarrow \{0,1\}$: *Given the group parameter* gp*, the group public key* gpk*, a message* msg *and a signature* σ*, the verifier runs this deterministic algorithm and outputs* 1 *if* σ *is a valid signature on* msg*, else outputs* 0*.*

$\mathsf{Open}(\mathsf{gp}, \mathsf{gpk}, \mathsf{opk}, msg, \sigma) \rightarrow (\mathsf{upk}, \mathsf{Prf})$: *This is a deterministic algorithm run by the* GM *which takes inputs the group parameter* gp*, the group public key* gpk*, the opening key* opk*, a message* msg *and a signature* σ*. The algorithm outputs the user public key* upk *and a proof* Prf *which ensures that the signature* σ *on the message* msg *is indeed generated by the user with public key* upk*. In case of* $\mathsf{upk} = \perp$*,* $\mathsf{Prf} = \perp$ *.*

$\mathsf{Judge}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma, (\mathsf{upk}, \mathsf{Prf})) \rightarrow \{0,1\}$: *This is a deterministic algorithm that takes inputs the group parameter* gp*, the group public key* gpk*, a message* msg*, a signature* σ*, a user public key* upk *and a proof* Prf *and outputs 1 if the proof* Prf *guarantees that the signature* σ *on the message* msg *is indeed generated by the user public key* upk*, else outputs 0.*

$\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, \mathsf{tr}) \rightarrow \{0,1\}$: *This is a deterministic algorithm run by the* GM *taking inputs the group parameter* gp*, the group public key* gpk*, a certificate* cert*, witness* w^{escrw}*, trace bit* tr *and outputs* 1 *if the witness confirms the choice of* tr*, else outputs* 0*.*

Correctness. For a traceable user ($\mathsf{tr} = 1$), an ATS scheme is said to be correct if for all security parameter λ, all $\mathsf{gp} \leftarrow \mathsf{Setup}(1^{\lambda})$, all $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{GrKeyGen}(\mathsf{gp})$, all $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{UsKeyGen}(\mathsf{gp})$, all $(\mathsf{cert}, w^{\mathsf{escrw}}) \leftarrow \mathsf{Enroll}(\mathsf{gp}, \mathsf{gpk}, \mathsf{isk}, \mathsf{upk}, \mathsf{tr} = 1)$, all $\sigma \leftarrow \mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, \mathsf{usk}, msg)$ it must hold that

$$\mathsf{Verify}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma) = 1$$
$$\mathsf{Judge}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma, \mathsf{Open}(\mathsf{gp}, \mathsf{gpk}, \mathsf{opk}, msg, \sigma)) = 1$$
$$\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, 1) = 1$$

For a non-traceable user ($\mathsf{tr} = 0$), an ATS scheme is said to be correct if for all security parameter λ, all $\mathsf{gp} \leftarrow \mathsf{Setup}(1^{\lambda})$, all $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{GrKeyGen}(\mathsf{gp})$, all $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{UsKeyGen}(\mathsf{gp})$, all $(\mathsf{cert}, w^{\mathsf{escrw}}) \leftarrow \mathsf{Enroll}(\mathsf{gp}, \mathsf{gpk}, \mathsf{isk}, \mathsf{upk}, \mathsf{tr} = 0)$, all $\sigma \leftarrow \mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, \mathsf{usk}, msg)$ it must hold that

$$\mathsf{Verify}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma) = 1$$
$$\mathsf{Open}(\mathsf{gp}, \mathsf{gpk}, \mathsf{opk}, msg, \sigma) = \perp$$
$$\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, 0) = 1$$

5.2 Security Model

The ATS scheme consists of the following five security requirements:
(i) *anonymity under tracing* (AuT) (ii) *traceability* (Trace) (iii) *non-frameability* (NF) (iv) *anonymity with accountability* (AwA) and (v) *trace-obliviousness* (TO).

(i) *Anonymity under tracing* (AuT): Anonymity under tracing requires that even a traceable user is anonymous to the adversary who is not having the opening key. This guarantees that only the GM is allowed to trace a traceable user. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{AuT}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 9.

```
$\mathsf{Exp}^{\mathsf{AuT}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$
$\mathsf{gp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{gpk},\ \mathsf{gsk} = (\mathsf{isk},\ \mathsf{opk})) \leftarrow \mathsf{GrKeyGen}(\mathsf{gp})$
$b' \leftarrow \mathcal{A}^{\mathsf{Ch}_1,\mathsf{Open}_1}(\mathsf{gp}, \mathsf{gpk}, \mathsf{isk})$
return $b'$

Oracle $\mathsf{Open}_1(\mathsf{gp},\ \mathsf{gpk},\ msg,\ \sigma)$
  if $\sigma \in Q$, then return $\perp$
  else return
  $(\mathsf{upk},\ \mathsf{Prf}) \leftarrow \mathsf{Open}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{opk},\ msg, \sigma)$

Oracle $\mathsf{Ch}_1(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{cert}_0,\ \mathsf{cert}_1,\ \mathsf{usk}_0,\ \mathsf{usk}_1,\ msg,\ w_0^{\mathsf{escrw}},\ w_1^{\mathsf{escrw}}, 1)$
$\sigma_0 \leftarrow \mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}_0, \mathsf{usk}_0, msg)$
$\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}_1, \mathsf{usk}_1, msg)$
if $(\sigma_0 \neq \perp \wedge \sigma_1 \neq \perp \wedge$
    $\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}_0, w_0^{\mathsf{escrw}}, 1) \wedge$
    $\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}_1, w_1^{\mathsf{escrw}}, 1)\ )$
    $Q \leftarrow Q \cup \{\sigma_b\}$
    return $\sigma_b$
else return $\perp$
```

Fig. 9. The anonymity under tracing experiment $\mathsf{Exp}^{\mathsf{AuT}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

Definition 19. *(Anonymity under tracing). An ATS scheme satisfies anonymity under tracing if the advantage of any PPT adversary $\mathcal{A}$ defined as $Adv^{AuT}_{ATS,\,\mathcal{A}}(\lambda)$ $= |Pr[\mathsf{Exp}^{AuT\text{ - }1}_{ATS,\,\mathcal{A}}(\lambda) = 1] - Pr[\mathsf{Exp}^{AuT\text{ - }0}_{ATS,\,\mathcal{A}}(\lambda) = 1]|$ is negligible.*

(ii) *Traceability* (Trace): This security attribute requires that every valid signature will trace to someone as long as the adversary does not hold both the certificate and user secret key of a non-traceable user. In the standard traceability game the GM can open any message and trace any of the user. In the case of the ATS scheme, when adversary queries certificate of a user of his choice, challenger will always generate certificate for a traceable user. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{Trace}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 10.

Definition 20. *(Traceability). An ATS scheme satisfies traceability if the advantage $Adv^{Trace}_{ATS,\,\mathcal{A}}(\lambda) = Pr[\mathsf{Exp}^{Trace}_{ATS,\,\mathcal{A}}(\lambda) = 1|$ of any PPT adversary $\mathcal{A}$ is negligible.*

(iii) *Non-frameability* (NF): Non-frameability necessitates that even though the GM and other users gets corrupted by the adversary, still they will not be able

```
$\mathsf{Exp}^{\mathsf{Trace}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$
$\mathsf{gp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{gpk},\, \mathsf{gsk} = (\mathsf{isk},\, \mathsf{opk})) \leftarrow \mathsf{GrKeyGen}(\mathsf{gp})$
$(msg,\, \sigma) \leftarrow \mathcal{A}^{\mathsf{UsKeyGen},\, \mathsf{Enroll}_1,\, \mathsf{Sign},\, \mathsf{Open}_2}(\mathsf{gp},\, \mathsf{gpk})$
return 0 if $(msg,\, \sigma) \in Q$ or
  $\mathsf{Verify}(\mathsf{gp},\, \mathsf{gpk},\, msg,\, \sigma) = 0$
else $(\mathsf{upk},\, \mathsf{Prf}) \leftarrow \mathsf{Open}(\mathsf{gp},\, \mathsf{gpk},\, \mathsf{opk},\, msg,\, \sigma)$
  return 1 if $\mathsf{upk} = \perp$ or
      $\mathsf{Judge}(\mathsf{gp},\, \mathsf{gpk},\, msg,\, \sigma,\, \mathsf{upk},\, \mathsf{Prf}) = 0$
  else return 0
Oracle $\mathsf{UsKeyGen}(\mathsf{gp})$
$(\mathsf{upk},\, \mathsf{usk}) \leftarrow \mathsf{UsKeyGen}(\mathsf{gp})$
$S[\mathsf{upk}] = \mathsf{usk}$
return $\mathsf{upk}$

Oracle $\mathsf{Enroll}_1(\mathsf{gp},\, \mathsf{gpk},\, \mathsf{upk},\, \mathsf{tr})$
let $\mathsf{tr}' = (\mathsf{upk} \notin \mathsf{dom}(S)) \in \{0, 1\}$
$(\mathsf{cert},\, w^{\mathsf{escrw}}) \leftarrow \mathsf{Enroll}(\mathsf{gp},\, \mathsf{gpk},\, \mathsf{isk},\, \mathsf{upk},\, \mathsf{tr} \vee \mathsf{tr}')$
return $\mathsf{cert}$
Oracle $\mathsf{Sign}(\mathsf{gp}, \mathsf{cert}, msg)$
$\mathsf{usk} = S[\mathsf{cert.upk}]$
if $(\mathsf{usk} = \perp)$, return $\perp$
else $\sigma \leftarrow \mathsf{Sign}(\mathsf{gp},\, \mathsf{gpk}, \mathsf{cert}, \mathsf{usk}, msg)$
     $Q = Q \cup \{(msg,\, \sigma)\}$
     return $\sigma$
Oracle $\mathsf{Open}_2(\mathsf{gp},\, \mathsf{gpk},\, msg, \sigma)$
$(\mathsf{upk},\, \mathsf{Prf}) \leftarrow \mathsf{Open}(\mathsf{gp},\, \mathsf{gpk}, \mathsf{opk}, msg, \sigma)$
return $(\mathsf{upk},\, \mathsf{Prf})$
```

Fig. 10. The traceability experiment $\mathsf{Exp}^{\mathsf{Trace}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

to sign messages on behalf of some honest user. Thus, traced signatures guarantee non-repudiation. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 11, where *st* is the state variable that stores the state information.

```
$\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

$\mathsf{gp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{gpk},\, st) \leftarrow \mathcal{A}(\mathsf{gp})$
if $\mathsf{gpk.gp} \neq \mathsf{gp}$, return $\perp$
$(msg,\, \sigma, \mathsf{upk},\, \mathsf{Prf}) \leftarrow \mathcal{A}^{\mathsf{UsKeyGen},\mathsf{Sign}}(st)$
return 1 if $((msg,\, \sigma) \notin Q \wedge$
   $\mathsf{Verify}(\mathsf{gp},\, \mathsf{gpk},\, msg,\, \sigma) = 1 \wedge$
   $\mathsf{upk} \in \mathsf{dom}(S) \wedge$
   $\mathsf{Judge}(\mathsf{gp},\, \mathsf{gpk},\, msg,\, \sigma\ \mathsf{upk},\, \mathsf{Prf}) = 1)$

Oracle $\mathsf{UsKeyGen}(\mathsf{gp})$
$(\mathsf{upk},\, \mathsf{usk}) \leftarrow \mathsf{UsKeyGen}(\mathsf{gp})$
$S[\mathsf{upk}] = \mathsf{usk}$
return $\mathsf{upk}$
Oracle $\mathsf{Sign}(\mathsf{gp},\, \mathsf{cert},\, msg)$
$\mathsf{usk} = S[\mathsf{cert.upk}]$
if $(\mathsf{usk} = \perp)$ return $\perp$
else $\sigma \leftarrow \mathsf{Sign}(\mathsf{gp},\, \mathsf{gpk},\, \mathsf{cert},\, \mathsf{usk},\, msg)$
  $Q = Q \cup \{(msg,\, \sigma)\}$
  return $\sigma$
```

Fig. 11. The non-frameability experiment $\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

Definition 21. *(Non-frameability). An ATS scheme satisfies non-frameability if the advantage $\mathsf{Adv}^{\mathsf{NF}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda) = 1|$ of any PPT adversary $\mathcal{A}$ is negligible.*

(iv) *Anonymity with accountability* (AwA): This security attribute requires that a non-traceable user remains anonymous even from a corrupted authority, inspite of the fact that the authority has full control over the system. Thus, a user is anonymous as long as the escrow key in the user's certificate is private. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{AwA}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 12, where *st* is the state variable that stores the state information.

Definition 22. *(Anonymity with accountability). An ATS scheme satisfies anonymity with traceability if the advantage of any PPT adversary $\mathcal{A}$ defined as $\mathsf{Adv}^{\mathsf{AwA}}_{\mathsf{ATS},\,\mathcal{A}}(\lambda) = |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{AwA}\text{ - }1}_{\mathsf{ATS},\,\mathcal{A}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}^{\mathsf{AwA}\text{ - }0}_{\mathsf{ATS},\,\mathcal{A}}(\lambda) = 1]|$ is negligible.*

```
$\underline{\mathsf{Exp}^{\mathsf{AwA}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)}$

$\mathsf{gp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{gpk}, st) \leftarrow \mathcal{A}(\mathsf{gp})$
if $\mathsf{gpk.gp} \neq \mathsf{gp}$, return $\perp$
$b' \leftarrow \mathcal{A}^{\mathsf{Ch}_1}(st)$
return $b'$
```

```
$\underline{\textbf{Oracle}\ \mathsf{Ch}_1(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{cert}_0,\ \mathsf{cert}_1,\ \mathsf{usk}_0,\ \mathsf{usk}_1,\ msg,\ w_0^{\mathsf{escrw}},\ w_1^{\mathsf{escrw}}, 0)}$
$\sigma_0 \leftarrow \mathsf{Sign}(\mathsf{gp},\ \mathsf{gpk}, \mathsf{cert}_0, \mathsf{usk}_0, msg)$
$\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}_1, \mathsf{usk}_1, msg)$
if $(\sigma_0 \neq \perp \wedge \sigma_1 \neq \perp \wedge$
    $\mathsf{Account}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{cert}_0, w_0^{\mathsf{escrw}}, 0)\ \wedge$
    $\mathsf{Account}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{cert}_1, w_1^{\mathsf{escrw}}, 0)\ )$
    return $\sigma_b$
else return $\perp$
```

Fig. 12. The anonymity with accountability experiment $\mathsf{Exp}^{\mathsf{AwA}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

(v) *Trace-obliviousness* (TO): Trace-obliviousness requires that no user will be able to determine if their anonymity is preserved or they are traceable. This is formally characterized by means of the experiment $\mathsf{Exp}^{\mathsf{TO}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ described in Fig. 13.

```
$\underline{\mathsf{Exp}^{\mathsf{TO}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)}$
$\mathsf{gp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{gpk},\ \mathsf{gsk} = (\mathsf{isk},\ \mathsf{opk})) \leftarrow \mathsf{GrKeyGen}(\mathsf{gp})$
$b' \leftarrow \mathcal{A}^{\mathsf{Ch}_2,\ \mathsf{Enroll}_2,\ \mathsf{Open}_3}(\mathsf{gp},\ \mathsf{gpk})$
return $b'$
$\underline{\textbf{Oracle}\ \mathsf{Enroll}_2(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{upk},\ \mathsf{tr})}$
$(\mathsf{cert},\ w^{\mathsf{escrw}}) \leftarrow \mathsf{Enroll}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{isk},\ \mathsf{upk},\ \mathsf{tr})$
return $\mathsf{cert}$
```

```
$\underline{\textbf{Oracle}\ \mathsf{Ch}_2(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{upk})}$
$(\mathsf{cert},\ w^{\mathsf{escrw}}) \leftarrow \mathsf{Enroll}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{isk},\ \mathsf{upk}, b)$
$U = U \cup \{\mathsf{upk}\}$
return $\mathsf{cert}$.
$\underline{\textbf{Oracle}\ \mathsf{Open}_3(\mathsf{gp},\ \mathsf{gpk}, msg, \sigma)}$
$(\mathsf{upk},\ \mathsf{Prf}) \leftarrow \mathsf{Open}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{opk}, msg, \sigma)$
if $\mathsf{upk} \in U$, then return $\perp$
else return $(\mathsf{upk},\ \mathsf{Prf})$
```

Fig. 13. The trace-obliviousness experiment $\mathsf{Exp}^{\mathsf{TO}\text{ - }b}_{\mathsf{ATS},\,\mathcal{A}}(\lambda)$

Definition 23. *(Trace-obliviousness). An ATS scheme satisfies trace-obliviousness if the advantage* $Adv^{TO}_{ATS,\,\mathcal{A}}(\lambda) = |Pr[\mathsf{Exp}^{TO\text{ - }1}_{ATS,\,\mathcal{A}}(\lambda) = 1] - Pr[\mathsf{Exp}^{TO\text{ - }0}_{ATS,\,\mathcal{A}}(\lambda) = 1]|$ *of any PPT adversary* $\mathcal{A}$ *is negligible.*

5.3 Accountable Tracing Signature from Isogenies

In this section we show the concrete construction of our ATS-scheme from isogenies. The main ingredients for our ATS scheme are: the $\mathsf{CSI\text{-}FiSh}$ signature scheme [BKV19] recalled in Section 2.4, our $\mathsf{CSIKOE\text{-}512}$ scheme described in Section 4 and a zero-knowledge argument system described in Section 2.3.

$\mathsf{Setup}(1^\lambda) \longrightarrow \mathsf{gp}$: A trusted authority runs this algorithm on input a security parameter λ, and performs the following steps to generate the group parameter gp.

- Chooses a large prime p of the form $p = 4\,l_1 l_2 \ldots l_n - 1$ where the l_i are small distinct odd primes with $n = 74$, $l_1 = 3$, $l_{73} = 373$, and $l_{74} = 587$.
- Selects the base elliptic curve $E_0 : y^2 = x^3 + x \in \mathsf{Ell}_p(\mathcal{O})$ over $\mathbb{F}_p$ with $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

- Sets the message space $\mathcal{M} = \{0,1\}^\lambda$ and the generator of the ideal class group $\mathcal{G} = \mathsf{Cl}(\mathcal{O})$ to be $\mathfrak{g} = < 3, \pi - 1 >$ with class number N where π is the Frobenius endomorphism.
- Sample two cryptographic hash functions $H' : \{0,1\}^* \to [-(S-1), (S-1)]^t$ and $H_k : \mathbb{F}_p \to \{0,1\}^\lambda$ where $t = \lambda / \log S$.
- Sets the public parameter $pp = (p,\ \mathfrak{g},\ N,\ E_0,\ M_C,\ t,\ S,\ H_k,\ H')$ where $M_C : \mathsf{Ell}_p(\mathcal{O}) \to \mathbb{F}_p$, a function that maps isomorphism classes of elliptic curve to its Montgomery coefficient.
- Samples two elements $[\mathfrak{a}] = [\mathfrak{g}^a]$ and $[\mathfrak{r}] = [\mathfrak{g}^r] \in \mathcal{G}$ for some $a,\ r \in \mathbb{Z}_N$. Computes the elliptic curves $E_1^{(0)} = [a]E_0$ and $E_2^{(0)} = [a+r]E_0$ and sets the public key $\mathsf{pk}^{(0)} = (E_1^{(0)},\ E_2^{(0)})$. Generates $crs \leftarrow \mathit{\Pi}.\mathsf{Setup}(1^\lambda)$ (See Section 2.3) and finally sets the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t,\ S, H_k, H'),$ $\mathsf{pk}^{(0)} = (E_1^{(0)},\ E_2^{(0)}),\ crs)$.

$\mathsf{GrKeyGen}(\mathsf{gp}) \to (\mathsf{gpk},\ \mathsf{gsk})$: The GM runs this randomized algorithm to generate the group public key gpk and group secret key gsk by performing the following steps:

- Extracting $\mathfrak{g}$, N, E_0 from $\mathsf{gp}.pp$, samples $S-1$ elements $[\mathfrak{m}_i] = [\mathfrak{g}^{m_i}] \in \mathcal{G}$ for some $m_i \in \mathbb{Z}_N$, computes the elliptic curve $E_i = [m_i]E_0$ for $i = 1, 2, \ldots, S-1$ and sets $vk = \{E_1,\ E_2, \ldots, E_{S-1}\}$.
- Samples two elements $[\mathfrak{b}] = [\mathfrak{g}^b]$ and $[\mathfrak{s}] = [\mathfrak{g}^s] \in \mathcal{G}$ for some $b,\ s \in \mathbb{Z}_N$, computes the elliptic curves $E_1^{(1)} = [b]E_0$ and $E_2^{(1)} = [b+s]E_0$, sets the public key $\mathsf{pk}^{(1)} = (E_1^{(1)},\ E_2^{(1)})$ and the secret key $\mathsf{sk}^{(1)} = s$. The GM publishes the group public key $\mathsf{gpk} = (\mathsf{gp} = (pp,\ \mathsf{pk}^{(0)} = (E_1^{(0)},\ E_2^{(0)}),\ crs),\ vk,\ \mathsf{pk}^{(1)} =$ $(E_1^{(1)}, E_2^{(1)}))$ and keeps the group secret key $\mathsf{gsk} = (\mathsf{isk} = (m_1, m_2, \ldots, m_{S-1}),$ $\mathsf{opk} = s)$ secret to himself.

$\mathsf{UsKeyGen}(\mathsf{gp}) \to (\mathsf{upk},\ \mathsf{usk})$: This is a randomized algorithm executed by a user that takes input the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t,\ S, H_k, H'),$ $\mathsf{pk}^{(0)} = (E_1^{(0)},\ E_2^{(0)}),\ crs)$ and generates its user public key upk and user secret key usk.

- Samples $S-1$ elements $[\mathfrak{n}_i] = [\mathfrak{g}^{n_i}] \in \mathcal{G}$ for some $n_i \in \mathbb{Z}_N$, computes the elliptic curve $E_i' = [n_i]E_0$ for $i = 1, 2, \ldots, S-1$ where $\mathfrak{g}$, N, E_0 are extracted from $\mathsf{gp}.pp$. Sets the user public key $\mathsf{upk} = \{E_1',\ E_2', \ldots, E_{S-1}'\}$ and the user secret key $\mathsf{usk} = (n_1, n_2, \ldots, n_{S-1})$.

$\mathsf{Enroll}(\mathsf{gp},\ \mathsf{gpk},\ \mathsf{isk},\ \mathsf{upk},\ \mathsf{tr}) \to (\mathsf{cert}, w^{\mathsf{escrw}})$: The GM runs this algorithm taking inputs the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t,\ S, H_k, H'),\ \mathsf{pk}^{(0)} =$ $(E_1^{(0)},\ E_2^{(0)}),\ crs)$, the group public key $\mathsf{gpk} = (\mathsf{gp},\ vk,\ \mathsf{pk}^{(1)} = (E_1^{(1)},\ E_2^{(1)}))$, a user public key $\mathsf{upk} = \{E_1',\ E_2', \ldots, E_{S-1}'\}$, an issue key $\mathsf{isk} = (m_1, m_2, \ldots, m_{S-1})$ and a value of trace bit $\mathsf{tr} \in \{0,1\}$. He produces a certificate - witness pair (cert, w^{escrw}) to the bit tr by performing the following steps:

– Randomizes the public key $\mathsf{pk}^{(\mathsf{tr})}$= $(E_1^{(\mathsf{tr})},\, E_2^{(\mathsf{tr})}))$ and generates a new public key epk by sampling $[\mathfrak{r}'] = [\mathfrak{g}^{r'}] \in \mathcal{G}$ for some $r' \in \mathbb{Z}_N$, computing the elliptic curve $E_1'^{(\mathsf{tr})} = [r']E_1^{(\mathsf{tr})}$ and $E_2'^{(\mathsf{tr})} = [r']E_2^{(\mathsf{tr})}$ and finally setting epk $= (E_1'^{(\mathsf{tr})},\, E_2'^{(\mathsf{tr})})$.

– Generates a $\mathsf{CSI\text{-}FiSh}$ signature σ_{cert} on $\mathsf{upk}||\mathsf{epk} = E_1'||\ldots||E_{S-1}'||E_1'^{(\mathsf{tr})}||E_2'^{(\mathsf{tr})}$ using the issue key isk= $(m_1, m_2, \ldots, m_{S-1})$ by setting $m_0 \leftarrow 0$, sampling $[\mathfrak{m}_i'] = [\mathfrak{g}^{m_i'}] \in \mathcal{G}$ for some $m_i' \in \mathbb{Z}_N$, computing t commitment elliptic curves $\widehat{E_i} = [m_i']E_0$ for $i = 1, 2, \ldots, t$ and generating the challenge string of length t over $[-(S-1),\, (S-1)]$ as follows:

$$(ch_1,\, ch_2, \ldots, ch_t) = H'(\widehat{E_1}||\widehat{E_2}||\ldots||\widehat{E_t}||\mathsf{upk}||\mathsf{epk}).$$

The GM computes the response $z_i = m_i'$ - $\mathsf{sgn}(ch_i)\, m_{|\,ch_i\,|} \pmod N$ using issue key isk and sets the signature $\sigma_{\mathsf{cert}} = (ch_1, ch_2, \ldots, ch_t, z_1, z_2, \ldots, z_t)$, where $\mathsf{sgn}(ch_i)$ denotes the sign of ch_i. He finally sends the certificate $\mathsf{cert} =$ $(\mathsf{upk} = \{E_1',\, E_2', \ldots, E_{S-1}'\}$, $\mathsf{epk} = (E_1'^{(\mathsf{tr})},\, E_2'^{(\mathsf{tr})})$, $\sigma_{\mathsf{cert}} = (ch_1, ch_2, \ldots, ch_t,$ $z_1, z_2, \ldots, z_t))$ to the user and keeps secret $w^{\mathsf{escrw}} = r'$.

$\mathsf{Sign}(\mathsf{gp},\, \mathsf{gpk},\, \mathsf{cert},\, \mathsf{usk},\, msg) \to \sigma$: This randomized algorithm is run by a user to generate a signature σ on a message $msg \in \mathcal{M}$ using the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t,\, S, H_k, H'),\, \mathsf{pk}^{(0)} = (E_1^{(0)},\, E_2^{(0)}),\, crs)$, the group public key $\mathsf{gpk} = (\mathsf{gp},\, vk,\, \mathsf{pk}^{(1)} = (E_1^{(1)},\, E_2^{(1)}))$, a user certificate $\mathsf{cert} = (\mathsf{upk} =$ $\{E_1',\, E_2', \ldots, E_{S-1}'\}, \mathsf{epk} = (E_1'^{(\mathsf{tr})},\, E_2'^{(\mathsf{tr})}), \sigma_{\mathsf{cert}} = (ch_1, ch_2, \ldots, ch_t, z_1, z_2, \ldots, z_t))$ and a user secret key $\mathsf{usk} = (n_1, n_2, \ldots, n_{S-1})$ in the following manner:

– Samples $S-1$ elements $[\mathfrak{e}_i] = [\mathfrak{g}^{e_i}] \in \mathcal{G}$ for some $e_i \in \mathbb{Z}_N$, computes the elliptic curve $E_i'' = [e_i]E_0$ for $i = 1, 2, \ldots, S-1$ and sets $pk = \{E_1'',\, E_2'', \ldots, E_{S-1}''\}$ and $sk = (e_1, e_2, \ldots, e_{S-1})$.

– Computes a signature σ_u on the message $pk = \{E_1'',\, E_2'', \ldots, E_{S-1}''\}$ using $\mathsf{usk} = (n_1, n_2, \ldots, n_{S-1})$ as the signing key. For which the user sets $n_0 \leftarrow 0$, samples $[\mathfrak{n}_i'] = [\mathfrak{g}^{n_i'}] \in \mathcal{G}$ for some $n_i' \in \mathbb{Z}_N$, computes t commitment elliptic curves $\widetilde{E_i} = [n_i']E_0$, the challenge string of length t over $[-(S-1),\, (S-1)]$ given by:

$$(ch_1',\, ch_2', \ldots, ch_t') = H'(\widetilde{E_1}||\widetilde{E_2}||\ldots||\widetilde{E_t}||pk),$$

the response $z_i' = n_i'$ - $\mathsf{sign}(ch_i')\, n_{|\,ch_i'\,|} \pmod N$ for $i = 1, 2, \ldots, t$ and sets the signature $\sigma_u = (ch_1', ch_2', \ldots, ch_t', z_1', z_2', \ldots, z_t')$.

– Encrypts the message $\gamma = \mathsf{bin}(M_C(E_1'))||\ldots||\mathsf{bin}(M_C(E_{S-1}'))||\mathsf{bin}(ch_1')||\ldots||$ $\mathsf{bin}(ch_t')||\mathsf{bin}(z_1')||\ldots||\mathsf{bin}(z_t')$ using randomized public key $\mathsf{epk} = (E_1'^{(\mathsf{tr})},\, E_2'^{(\mathsf{tr})})$ extracted from cert to generate the ciphertext $ct = (ct_1,\, ct_2)$. For which the user samples $[\mathfrak{q}] = [\mathfrak{g}^q] \in \mathcal{G}$ for some $q \in \mathbb{Z}_N$ and sets $ct_1 = [q]E_1'^{(\mathsf{tr})}$, $ct_2 = H_k(M_C([q]E_2'^{(\mathsf{tr})})) \oplus \gamma$ and the encryption randomness $rand = q$. Note that this encryption is of the $\mathsf{CSIKOE\text{-}512}$ scheme described in section 4.

– Generates a proof $\pi \leftarrow \mathit{\Pi}.\mathsf{Prove}(crs, \mathsf{stmt}, \mathsf{wit})$ (See Section 2.3) using crs for the following relation $\mathcal{R}$ to prove knowledge of $(\mathsf{upk}, \mathsf{epk}, \sigma_{\mathsf{cert}}, \sigma_u)$ where the statement $\mathsf{stmt} = (ct = (ct_1, ct_2), pk = \{E''_1, E''_2, \dots, E''_{S-1}\}, vk = \{E_1, E_2 \dots, E_{S-1}\})$ and witness $\mathsf{wit} = (\mathsf{upk} = \{E'_1, E'_2, \dots, E'_{S-1}\}, \mathsf{epk} = ({E'_1}^{(\mathsf{tr})}, {E'_2}^{(\mathsf{tr})}), \sigma_{\mathsf{cert}} = (ch_1, ch_2, \dots, ch_t, z_1, z_2, \dots, z_t), \sigma_u = (ch'_1, ch'_2, \dots, ch'_t, z'_1, z'_2, \dots, z'_t), rand = q)$. We say that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$ if and only if the following three relations hold:

1. The ciphertext $ct = (ct_1, ct_2)$ must be a correct encryption of message $\gamma = \mathsf{bin}(M_C(E'_1))|| \dots ||\mathsf{bin}(M_C(E'_{S-1}))||\mathsf{bin}(ch'_1)|| \dots ||\mathsf{bin}(ch'_t)||\mathsf{bin}(z'_1)|| \dots ||\mathsf{bin}(z'_t)$ under the public key epk with encrytion randomness $rand = q$ satisfying:
$$ct_1 = [q]{E'_1}^{(\mathsf{tr})},\ ct_2 = H_k(M_C([q]{E'_2}^{(\mathsf{tr})})) \oplus \gamma$$
2. The $\mathsf{CSI\text{-}FiSh}$ signature σ_u on the message pk must be a valid signature under the verification key upk satisfying
$$(ch'_1, ch'_2, \dots, ch'_t) = H'(\widetilde{E_1}||\widetilde{E_2}|| \dots ||\widetilde{E_t}||\, pk)$$
where $\widetilde{E_i} = [n'_i]E_0$ is recovered by computing $[z'_i]E'_{ch'_i}$ for $i = 1, 2, \dots, t$.
3. The $\mathsf{CSI\text{-}FiSh}$ signature σ_{cert} on $\mathsf{upk}||\mathsf{epk}$ must be a valid signature under the verification key vk satisfying:
$$(ch_1, ch_2, \dots, ch_t) = H'(\widehat{E_1}||\widehat{E_2}|| \dots ||\widehat{E_t}||\mathsf{upk}||\mathsf{epk})$$
where $\widehat{E_i} = [m'_i]E_0$ is recovered by computing $[z_i]E_{ch_i}$ for $i = 1, 2, \dots, t$.

– Generates a $\mathsf{CSI\text{-}FiSh}$ signature σ_0 on the message $msg \,||\, ct \,||\, pk \,||\, vk \,||\, \pi$, i.e.,
$$msg||ct_1||ct_2||E''_1|| \dots ||E''_{S-1}||E_1|| \dots ||E_{S-1}||\pi$$
taking $sk = (e_1, e_2, \dots, e_{S-1})$ as the signing key. For which the user sets $e_0 \leftarrow 0$, samples $[\mathfrak{e}'_i] = [\mathfrak{g}^{e'_i}] \in \mathcal{G}$ for some $e'_i \in \mathbb{Z}_N$. Computes t commitment elliptic curves $\overline{E_i} = [e'_i]E_0$, the challenge string of length t over $[-(S-1), (S-1)]$ given by:
$$(ch''_1, ch''_2, \dots, ch''_t) = H'(\overline{E_1}|| \dots ||\overline{E_t}||\, msg||ct||pk||vk||\pi)$$
and the response $z''_i = e'_i$ - $\text{sign}(ch''_i)\, e_{|\, ch''_i\,|} \pmod N$ for $i = 1, 2, \dots, t$. Sets the signature $\sigma_0 = (ch''_1, ch''_2, \dots, ch''_t, z''_1, z''_2, \dots, z''_t)$.

– Finally, outputs $\sigma = (\sigma_0, pk, ct = (ct_1, ct_2), \pi)$ as the signature on the message $msg \in \mathcal{M}$.

$\mathsf{Verify}(\mathsf{gp}, \mathsf{gpk}, msg, \sigma) \rightarrow \{0, 1\}$: This is a deterministic algorithm that verifies the signature $\sigma = (\sigma_0 = (ch''_1, ch''_2, \dots, ch''_t, z''_1, z''_2, \dots, z''_t), pk = \{E''_1, \dots, E''_{S-1}\}, ct = (ct_1, ct_2), \pi)$ on the message $msg \in \mathcal{M}$ by performing the following steps using the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t, S, H_k, H'), \mathsf{pk}^{(0)} = (E_1^{(0)}, E_2^{(0)}), crs)$ and the group public key $\mathsf{gpk} = (\mathsf{gp}, vk, \mathsf{pk}^{(1)} = (E_1^{(1)}, E_2^{(1)}))$.

– Parse $\sigma_0 = (ch''_1, ch''_2, \ldots, ch''_t, z''_1, z''_2, \ldots, z''_t)$. Defines $E_{-i} = E_i^t$ for $i = 1, 2, \ldots, S-1$, where E_i^t is the twist[1] of the elliptic curve E_i. Recovers t elliptic curves $\overline{E_i} = [z''_i]E_{ch''_i}$ for $i = 1, 2, \ldots, t$. If $(ch''_1, ch''_2, \ldots, ch''_t) = H'(\overline{E_1}||\ldots||\overline{E_t}||msg||ct||pk||vk||\pi)$ returns 1, indicating σ_0 is a valid CSI-FiSh signature on $msg||ct||pk||vk||\pi$ under $sk = (e_1, e_2, \ldots, e_{S-1})$, else returns 0.

– Runs Π.Verify(crs, stmt, π) (See Section 2.3) taking input the statement stmt $= (ct, pk, vk)$ and crs to verify the proof π, where pk, ct are extracted from σ and vk is obtained from gpk. If all the checks succeed, returns 1, else returns 0.

Open(gp, gpk, opk, msg, σ) $\rightarrow$ (upk, Prf): This is a deterministic algorithm run by the GM which takes inputs the group parameter gp $= (pp = (p, \mathfrak{g}, N, E_0, M_C, t, S, H_k, H')$, pk$^{(0)} = (E_1^{(0)}, E_2^{(0)})$, crs), the group public key gpk= (gp, vk, pk$^{(1)} = (E_1^{(1)}, E_2^{(1)})$), the opening key opk $= s$, a message $msg \in \mathcal{M}$ and a signature $\sigma = (\sigma_0, pk, ct = (ct_1, ct_2), \pi)$ and outputs the user public key upk and a proof Prf in the following manner:

– Runs Verify(gp, gpk, msg, σ) and aborts if it fails.

– Extracts the ciphertext $ct = (ct_1, ct_2)$ from the signature σ and recovers the message $\gamma = \mathsf{bin}(M_C(E'_1))||\ldots||\mathsf{bin}(M_C(E'_{S-1}))||\mathsf{bin}(ch'_1)||\ldots||\mathsf{bin}(ch'_t)||$ $\mathsf{bin}(z'_1)||\ldots||\mathsf{bin}(z'_t)$ using the opening key opk $= s$ by evaluating $ct_2 \oplus H_k(M_C([s]ct_1))$ and finally computing upk $= \{E'_1, E'_2, \ldots, E'_{S-1}\}$ and $\sigma_u = (ch'_1, ch'_2, \ldots, ch'_t, z'_1, z'_2, \ldots, z'_t)$ from γ.
Note that, $ct_2 \oplus H_k(M_C([s]ct_1)) = \gamma$, which follows from the correctness of CSIKOE-512.

– Outputs upk $= \{E'_1, E'_2, \ldots, E'_{S-1}\}$ and Prf $= \sigma_u$.

Judge(gp, gpk, upk, Prf, msg, σ) $\rightarrow \{0, 1\}$: This is a deterministic algorithm that takes inputs the group parameter gp $= (pp = (p, \mathfrak{g}, N, E_0, M_C, t, S, H_k, H')$, pk$^{(0)} = (E_1^{(0)}, E_2^{(0)})$, crs), the group public key gpk $=$ (gp, vk, pk$^{(1)} = (E_1^{(1)}, E_2^{(1)})$), a message msg, a signature $\sigma = (\sigma_0, pk, ct = (ct_1, ct_2), \pi)$, a user public key upk $= \{E'_1, E'_2, \ldots, E'_{S-1}\}$ and a proof Prf $= \sigma_u$ and outputs 0 or 1 by executing the below steps:

– Runs Verify(gp, gpk, msg, σ) and aborts if it fails.

– Parse $\sigma_u = (ch'_1, ch'_2, \ldots, ch'_t, z'_1, z'_2, \ldots, z'_t)$. Defines $E_{-i} = E_i^t$ for $i = 1, 2, \ldots, S-1$, where E_i^t is the twist of the elliptic curve E_i. Recovers t elliptic curves $\widetilde{E_i} = [n'_i]E_0$ by computing $[z'_i]E'_{ch'_i}$ for $i = 1, 2, \ldots, t$. If $(ch'_1, ch'_2, \ldots, ch'_t) = H'(\widetilde{E_1}||\ldots||\widetilde{E_t}||pk)$ returns 1, indicating σ_u is a valid CSI-FiSh signature on pk under the user secret key usk $= (n_1, n_2, \ldots, n_{S-1})$ or else returns 0.

[1] The quadratic twist of an elliptic curve $E : y^2 = f(x)$ defined over a field K is given by $E^t : dy^2 = f(x)$ where $d \in K$ has Legendre symbol value -1.

– If all the checks succeed returns 1, or else returns 0.

$\mathsf{Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, \mathsf{tr}) \to \{0,1\}$: This is a deterministic algorithm run by the GM taking inputs the group parameter $\mathsf{gp} = (pp = (p, \mathfrak{g}, N, E_0, M_C, t,\ S, H_k, H'), \mathsf{pk}^{(0)} = (E_1^{(0)},\ E_2^{(0)}), crs)$, the group public key $\mathsf{gpk} = (\mathsf{gp}, vk, \mathsf{pk}^{(1)} = (E_1^{(1)}, E_2^{(1)}))$, a certificate $\mathsf{cert} = (\mathsf{upk} = \{E'_1,\ E'_2, \ldots, E'_{S-1}\},\ \mathsf{epk} = ({E'_1}^{(\mathsf{tr})},\ {E'_2}^{(\mathsf{tr})}), \sigma_{\mathsf{cert}} = (ch_1, ch_2, \ldots, ch_t, z_1, z_2, \ldots, z_t))$, witness $w^{\mathsf{escrw}} = r'$, trace bit tr and checks if the equality $([r']E_1^{(\mathsf{tr})},\ [r']E_2^{(\mathsf{tr})}) = ({E'_1}^{(\mathsf{tr})},\ {E'_2}^{(\mathsf{tr})})$ holds. If the verification succeeds return 1, else return 0.

Correctness. The correctness of our ATS scheme is described as follows:

1. For any honestly generated signature σ by a user on a message $msg \in \mathcal{M}$ the $\mathsf{ATS.Verify}$ algorithm will output 1 with probability 1. This follows from the correctness of the $\mathsf{CSI\text{-}FiSh}$ signature scheme and completeness of the zero-knowledge argument system.
2. For an honest traceable user, $\mathsf{ATS.Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, 1)$ algorithm will output 1 with probability 1. This is because the check $([r']E_1^{(1)},\ [r']E_2^{(1)}) = ({E'_1}^{(1)},\ {E'_2}^{(1)})$ succeeds for an honest GM. $\mathsf{ATS.Open}$ algorithm recovers the true signer and outputs a valid proof which follows from the correctness of the $\mathsf{CSI\text{-}FiSh}$ signature scheme and the correctness of our $\mathsf{CSIKOE\text{-}512}$ scheme described in section 4. The $\mathsf{ATS.Judge}$ algorithm accepts the proof given by the $\mathsf{ATS.Open}$ algorithm with probability 1 due to the completeness of zero-knowledge argument system and the correctness of the $\mathsf{CSI\text{-}FiSh}$ signature scheme.
3. For an honest non-traceable user, $\mathsf{ATS.Account}(\mathsf{gp}, \mathsf{gpk}, \mathsf{cert}, w^{\mathsf{escrw}}, 0)$ algorithm will output 1 as the check $([r']E_1^{(0)},\ [r']E_2^{(0)}) = ({E'_1}^{(0)},\ {E'_2}^{(0)})$ succeeds for an honest group manager. $\mathsf{ATS.Open}$ algorithm outputs $\perp$ as opening key $\mathsf{opk} \neq r$ with overwhelming probability. Thus the decryption algorithm with not be able to retrieve the plaintext; consequently the real signer remains anonymous.

Efficiency. Since our ATS scheme is the first isogeny based accountable tracing signature scheme, we do not compare the efficiency of our scheme with other works. The key and signature size of our scheme grows as S grows and thus it is not very reasonable. But that can be reduced somewhat using the Merkle tree technique and other optimizations stated in [BKV19]. From the efficiency point of view, our ATS scheme is not up to the mark and needs a lot more optimization. However, we firmly believe that it will open avenues for more research in this direction.

The following theorem follows from Theorem 12 of [KM15].

Theorem 13. *Under the assumption that $\mathsf{CSI\text{-}FiSh}$ signature scheme described in Section 2.4 is strongly unforgeable, $\mathsf{CSIKOE\text{-}512}$ scheme described in Section 4 satisfies key randomizability, plaintext indistinguishability under key ran-*

domization and key privacy under key randomization and Π is zero-knowledge simulation-extractable argument system as defined in Section 2.3, the isogeny based ATS *scheme presented in Section 5.3 satisfies anonymity under tracing, traceability, non-frameability, anonymity with accountability and trace-obliviousness as per definitions in Section 5.2.*

References

BKV19. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: efficient isogeny based signatures through class group computations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 227–247. Springer, 2019.

CLM$^+$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*, pages 395–427. Springer, 2018.

Cou06. Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptol. ePrint Arch.*, 2006:291, 2006.

CSV20. Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional diffie-hellman problem for class group actions using genus theory. In *Annual International Cryptology Conference*, pages 92–120. Springer, 2020.

DFG19. Luca De Feo and Steven D Galbraith. Seasign: Compact isogeny signatures from class group actions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 759–789. Springer, 2019.

DFM20. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In *IACR International Conference on Public-Key Cryptography*, pages 187–212. Springer, 2020.

KM15. Markulf Kohlweiss and Ian Miers. Accountable metadata-hiding escrow: A group signature case study. *Proceedings on Privacy Enhancing Technologies*, 2015(2):206–221, 2015.

LNWX19. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Accountable tracing signatures from lattices. In *Cryptographers' Track at the RSA Conference*, pages 556–576. Springer, 2019.

MOT20. Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. Sigamal: A supersingular isogeny-based pke and its application to a prf. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 551–580. Springer, 2020.

Sho99. Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

Sil09. Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009.

Vél71. Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.

Wat69. William C Waterhouse. Abelian varieties over finite fields. In *Annales scientifiques de l'École Normale Supérieure*, volume 2, pages 521–560, 1969.