Compact, Efficient and UC-Secure Isogeny-Based Oblivious Transfer

Yi-Fu Lai[1], Steven D. Galbraith[1], and Cyprien Delpech de Saint Guilhem[2,3]

[1]*University of Auckland, New Zealand*
[2]*imec-COSIC, KU Leuven, Belgium*
[3]*Dept Computer Science, University of Bristol, United Kingdom*
ylai276@aucklanduni.ac.nz, s.galbraith@auckland.ac.nz, ,
cyprien.delpechdesaintguilhem@kuleuven.be

Abstract

Oblivious transfer (OT) is an essential tool of cryptographic protocols. It can serve as a building block for realizing all multiparty functionalities. The strongest security notion against malicious adversaries is universal composibility (UC-secure). Due to the rigorous algebraic structures and operations, achieving the specific security notion with isogenies is believed to be difficult. Hence, it is an open problem to have an efficient UC-secure OT oblivious transfer scheme based on isogenies.

In this work, we propose the first isogeny-based UC-secure oblivious transfer protocol in the presence of malicious adversaries without analogues in the Diffie-Hellman setting. The simple and compact CSIDH-based scheme consists of a constant number of isogeny computations. The underlying relaxed problem is called the computational reciprocal CSIDH problem which we can prove equivalent to the computational CSIDH problem with a quantum reduction.

1 Introduction

Oblivious transfer (OT) was first introduced by Rabin to establish the exchange of secrets protocol [Rab05] in 1981, based on the factoring problem. Say the sender has two messages, oblivious transfer allows the receiver to know one of them and keeps the sender oblivious to which message has been received while the unchosen message remains unknown to the receiver.

It has been shown that oblivious transfer is an important building block as a cryptographic tool. Oblivious transfer can be used to construct other primitives, like millionaires' problem protocols [IG03], or signing contracts protocols [EGL85], or, more generally, private multiparty computation protocols [CvdGT95] or even secure generic multiparty functionalities [Ode09]. Several oblivious transfer protocols based on Diffie-Hellman-related problems were proposed [BM89, NP01, PVW08, CO15, BDD+17].

Oblivious transfer protocols exist for various hardness assumptions. However, cryptographic protocols based on problems subordinate to either the discrete logarithm problem or the factoring problem will suffer a polynomial-time quantum attack, Shor's algorithm [Sho99]. Several post-quantum oblivious transfer schemes have been proposed, including Peikert et al.'s lattice-based OT [PVW08], and code-based OTs [DvdGMN08, DNMQ12, BDD+17]. Recently, some isogeny-based OTs have been independently proposed [BOB18, dSGOPS18, Vit19].

Concerning security of oblivious transfer, traditional security definitions aim at guaranteeing privacy for both parties, including one-sided simulation and the view-based definition for a two-message protocol [NP01, DvdGMN08], see Section 2.6 of [HL10]. These notions ensure privacy for both parties only in a standalone setting. However, in real world deployment, protocols are always composed into an enormous and complex construction. For security after composing, more and more oblivious transfer proposals [Lin08, PVW08, DNMQ12, BDD+17] follow the real/ideal paradigm and universally-composable security (UC security) provided by Canetti [Can01]. The impossibility results for some protocols have been given in [CKL03].

The first isogeny-based cryptosystem was proposed by Couveignes [Cou06], which included a key exchange scheme based on a hard homogeneous space. However, the paper was not published at that time. It was independently rediscovered by Rostovtsev and Stolbunov in 2006 [RS06]. Then, Jao and De Feo

proposed the Supersingular Isogeny Diffie Hellman (SIDH) in 2011 [JDF11]. Later, it was transformed into the Supersingular Isogeny Key Encapsulation (SIKE) [JAC+17] which included a public key encryption scheme and a key encapsulation mechanism and is now one of the third-round alternate candidates in the post-quantum cryptography standardization competition led by NIST [NIS20]. Castryck et al. devised an efficient key exchange cryptosystem called commutative SIDH (CSIDH) [CLM+18]. Castryck et al.'s setting of CSIDH-512 is conjectured to achieve NIST post-quantum security category 1 with the public key size of only 64 bytes [CLM+18] without any pseudorandom function key compression technique, which is far less in comparison with the candidates in the NIST competition [NIS20]. In this work, we exploit the structure of CSIDH to construct our scheme.

In comparison with the DH-based protocols, due to the algebraic structures, it is arguably more challenging to develop isogeny-based cryptosystems achieving the desired notion. For example, neither the randomizing procedure $(g^s y^t, x^s z^t) \leftarrow RAND(g, x, y, z; s, t \leftarrow_{\$} \mathbb{F}_p)$ [PVW08, Lin08] nor the fundamental one-trapdoor setup $pk_0 pk_1 = c$ where c is a constant in [NP01, BDD+17] can be transplanted to the isogeny-based setting.

We review the aforementioned isogeny-based oblivious transfer proposals [BOB18, dSGOPS18, Vit19]. Their schemes can be viewed as "tweaks of two Diffie-Hellman key exchange agreements" or variants of the Diffie-Hellman problem as stated in [CO15]. As a result, wider transmitting bandwidth, more public curves, or additional mechanism will serve as trade-offs for the constructions. Besides, as stated in [CO15], these schemes, including their scheme, cannot achieve full-simulatablility, even in the sense of the sequential composition theorem [Can00]. This is because a simulator cannot extract the input of the malicious adversary who delays the decryption and just stores everything. A natural solution is embedding a zero-knowledge proof protocol to extract the input of the adversary in the hybrid model but sacrificing efficiency. Chou and Orlandi [CO15] pointed out another potential solution that the receiver should show a "proof of timely decryption" while not leaking the secret input, which was realized by Barreto et al. in their framework in the updated version of [BDD+17].

On the aspect of security, the schemes [BOB18, dSGOPS18, Vit19] are all only proved secure under either semi-honest models or a non-simulation-based definition. In other words, the schemes all ensure nothing when executed within a larger environment against malicious adversaries. Regarding the underlying computational assumption, a reduction to a well-known one is a preferred choice since a reduction to a weaker variant may incur unpredictable flaws for the scheme. For instance, the scheme in [Vit19], as stated in their work, will be broken by malicious adversaries in CSIDH setting. Consequently, an isogeny-based oblivious transfer protocol proved to be secure under an assumption equivalent to a well-known problem and with respect to a powerful security notion is still currently missing from the literature.

1.1 Contributions

To resolve all issues presented above, this paper presents the first construction for a UC-secure isogeny-based oblivious transfer protocol in the presence of malicious adversaries with a variety of novel techniques. The construction is not only compact with a constant number of isogeny computations (see Table 1) but also a robust scheme without compromising the hardness. Our schemes use a feature that is available for isogenies (the quadratic twist) that does not have an analogue in the DLP setting, but some techniques are not limited to isogeny-based cryptography, see Section 6.

Firstly, we design a novel 1-out-of-2 oblivious transfer protocol by a small change to the Diffie-Hellman protocol to achieve a compact OT prototype with a trusted setup curve (or a public curve). Next, the 3-round protocol is transformed into a 2-round scheme through a new use of quadratic twists. The 2-round scheme is the most efficient isogeny-based OT scheme in the semi-honest model so far, see Section 5. Based on the modification, a secure mechanism can be established in which the receiver will demonstrate the "ability to decrypt" to the sender for one-side simulation, which is based on a similar idea of [CO15, BDD+17] but with a different mechanism. Furthermore, we replace the trusted setup with a trapdoor algorithm to accomplish the fully-simulatable construction. Finally, we introduce a new assumption, the reciprocal CSIDH problem, (Problem 5) that looks non-standard, but we prove equivalence to the computational CSIDH problem with a quantum reduction with a proposition we developed called "self-reconciling" (Proposition 2.2).

As pointed out by Canetti et al. [CKL03], it is impossible to achieve a UC-secure OT scheme without any trusted setup. Our construction is proposed in a hybrid model with two functionalities, see Table 2 for comparison with the related works concerning the hybrid models.

This paper is organized as follows. Section 2 briefly describes CSIDH, the related functionalities, as well as simulation-based definition for two-party protocols. Section 3 introduces how we construct our oblivious transfer protocol. Section 4 discusses the underlying assumptions against the semi-honest

adversary and then the malicious adversary. Next we will compare our OT with previous three isogeny-based OT protocols in section 5 and then conclude in section 6. For comprehensibility, the content related to the isogenies will frequently be accompanied or introduced by the counterparts in the Diffie-Hellman setting at the beginning.

1.2 Related work

There are three aforementioned isogeny-based OT protocols. All the adversary models are either semi-honest or non-simulation, which are both quite weak notions. While the semi-honest model cannot reflect vicious attackers in the real world, the non-simulation-based model cannot enjoy the composition theorems [Can00, Can01], see section 5.

The first protocol was proposed by Barreto et al. [BOB18] based on the common reference string (CRS) model along with the random oracle model. They revisited Chou and Orlandi's work [CO15] and proposed a SIDH-based OT. They exploited the property of SIDH to mask one party's public points by randomly (up to the receiver's choice) adding shared selective points derived from the common reference string. To achieve this, the implementation requires an additional coin-flipping mechanism while the security is only guaranteed in the semi-honest model.

The second one was proposed by de Saint Guilhem et al. [dSGOPS18]. They derived their two constructions from the Shamir-3-Pass key transport scheme and [CO15], respectively. Their framework is UC-secure against semi-honest adversaries based on a masking structure hard homogeneous space or on $\mathbb{F}_{p^2}$ supersingular isogenies.

The third one was proposed by Vitse [Vit19]. It is derived from Wu, Zhang, and Wang's OT [WZW03] based on a Diffie-Hellman-related problem. Their proposal naturally fits well in the general setting (including DH, SIDH, and CSIDH). They gave a game-based security definition (semantic security) for their OT protocol and gave the hardness proof of their assumption in generic groups.

2 Preliminaries

Notation.

Let $\{X(a,n)\} =_c \{Y(a,n)\}$ denote computationally indistinguishable probabilistic ensembles X, Y for which any PPT non-uniform algorithm there exists a non-negligible function f such that for all $a, n \in \mathbb{N}$ we have $|\Pr[D(X(a,n))] - \Pr[D(Y(a,n))]| \leq f(n)$. The notation $a \leftarrow_{\$} S$ means a is uniformly generated from the set S. For simplicity, we often omit the security level parameter n but it is implicit in the indistinguishabiliy and the negligible function.

2.1 CSIDH

For any given prime p and an elliptic curve E defined over $\mathbb{F}_p$, $End_p(E)$ is the subring of endomorphism ring $End(E)$ collecting the endomorphisms defined over $\mathbb{F}_p$.

Let $\mathcal{O}$ be an order in an imaginary quadratic field and $\pi \in \mathcal{O}$ an element of norm p. Define the set of isomorphism classes of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ where E defined over $\mathbb{F}_p$, $End_p(E) = \mathcal{O}$, and π is the $\mathbb{F}_p$-Frobenius map of E. For any ideal $\mathfrak{a} \in \mathcal{O}$ and $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, an action can be defined by $\mathfrak{a} * E = E'$ for an isogeny $\phi : E \to E'$ with $ker(\phi) = \cap_{\alpha \in \mathfrak{a}}\{P \in E(\bar{\mathbb{F}}_p) \mid \alpha(P) = 0\}$. The image curve of $\mathfrak{a} * E$ is well-defined up to isomorphisms. Moreover, the ideal class group $Cl(\mathcal{O})$ acts freely and transitively on $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$.

Castryck et al specified the prime to be $p = 4 \times \ell_1 \times ... \times \ell_n - 1$ where ℓ_i are small odd primes. In the case of $p = 3 \mod 8$, for any supersingular elliptic curve E defined over $\mathbb{F}_p$, the restricted endomorphism ring $End_p(E) = \mathbb{Z}\{\pi\} \cong \mathbb{Z}\{\sqrt{-p}\}$ if and only if E is $\mathbb{F}_p$-isomorphic to $E_A : y^2 = x^3 + Ax^2 + x$ for some unique $A \in \mathbb{F}_p$. The quadratic twist of a given elliptic curve $E : y^2 = f(x)$ is $E^t : dy^2 = f(x)$ where $d \in \mathbb{F}_p$ has Legendre symbol value -1. When $p = 3 \mod 4$ and $j(E) = 1728$, E and E^t are $\mathbb{F}_p$-isomorphic. The quadratic twist can be efficiently computed in the CSIDH setting [CLM+18]. Since the prime $p = 3 \mod 4$, $E' : -y^2 = x^3 + Ax^2 + x$ is the quadratic twist of $E_A : y^2 = x^3 + Ax^2 + x$ and E' is $\mathbb{F}_p$-isomorphic to E_{-A} by $(x, y) \mapsto (-x, y)$. Besides, the connected components of these special curves are symmetric. That is, $(a * E_0)^t = a^{-1} * E_0$. Therefore, for any curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, we have, by the transitivity of the action,

$$(a * E)^t = a^{-1} * E^t.$$

Throughout this paper, we concentrate on supersingular curves defined over $\mathbb{F}_p$. Denote the ideal class group $Cl(End_p(E))$ by Cl and the set of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ by $\mathcal{E}$. In CSIDH [CLM+18], the sampling

method of the class group Cl they provided is heuristically assumed to be uniform. Here we assume the sampling of elements in Cl is indistinguishable from the uniform distribution. A method closer to the uniform sampling can instantiate this by expanding the n-tuple of the CSIDH sampling from a larger integer space $[-B, B]$ to approximate the uniformity where a n-tuple $(e_i)_i$ represents ideal class $\prod_i (\ell_i, \pi - 1)^{e_i}$. Beullens et al. proposed a more efficient instantiation in CSI-FiSh [BKV19] that requires a pre-processing to complete a lattice of relations in the class group. Good references for the implementations of CSIDH are [CLM+18, BdFLS20] as well as the constant-time implementations [MCR19, OAYT19].

Next, We define some computational assumptions we need in this paper.

Problem 1. (Computational CSIDH Problem) Given curves E, $r * E$ and $s * E$ where $r, s \in Cl$. Then find $E' \in \mathcal{E}$ such that $E' = rs * E$.

Problem 2. Given curves $(E, s * E, r * E)$ where $r, s \in Cl$. Then find $E' \in \mathcal{E}$ such that $E' = s^{-1}r * E$.

The computational CSIDH problem is the main hardness assumption for [CLM+18]. Problem 2 is the equivalent problem. To see this, given an oracle O for Problem 1, one can obtain E' by taking $E' \leftarrow O(s * E, E, r * E)$ such that $E' = rs^{-1} * E$. Conversely, given an oracle O for Problem 2, one can obtain E' by taking $E' \leftarrow O(s * E, E, r * E)$ such that $E' = rs * E$.

The following two problems are the main underlying problems against semi-honest adversaries.

Problem 3. (Computational Square CSIDH Problem) Given curves E and $s * E$ where $s \in Cl$. Then find $E' \in \mathcal{E}$ such that $E' = s^2 * E$.

Problem 4. (Computational Inverse CSIDH Problem) Given curves E and $s * E$ where $s \in Cl$. Then find $E' \in \mathcal{E}$ such that $E' = s^{-1} * E$.

These problems are similar to the computational square Diffie-Hellman problem and the computational inverse Diffie-Hellman problem, respectively. That is, given (g, g^s), the challenge is to find g^{s^2} or $g^{s^{-1}}$. In the Diffie-Hellman version, these two problems are equivalent to the computational Diffie-Hellman problem (see Chapter 21 of [Gal12]). As Castryck et al. pointed out [CLM+18] both problems contain exceptional cases when E_0 takes part in the problems due to the symmetric structure. That is, $(a * E_0)^t = a^{-1} * E_0$. The issue can be circumvented if the public curve is generated by a trusted third party.

In the CSIDH setting, one can also show these two problems are equivalent. Given an oracle O of Problem 3 and $E, s * E$, one can obtain E' which satisfies $E' = s^{-1} * E$ with a oracle query $E' \leftarrow O(s * E, E)$. Conversely, given an oracle O of Problem 4 and $E, s * E$, one can obtain $E' = s^2 * E$ with a oracle query $E' \leftarrow O(s * E, E)$. Besides, one can also easily observe that they are not harder than the computational CSIDH problem by simply taking $r = s \in Cl$.

Regarding the converse reduction, even though finding the order of a class group is classically non-trivial, notice that class number can be found with a polynomial-time quantum algorithm [Sho99, Hal05]. We will show an efficient reduction to the computational CSIDH problem with the given class number.

Lemma 2.1. *Given $(E, a * E)$. Then for $n \in \mathbb{N}$ one can compute $a^n * E$ with given access to the oracle O for the square CSIDH problem with $O(\log(n))$ queries.*

Proof. At a glance, the double and add method seems not to work. But it indeed works based on the idea of the Montgomery ladder [Mon87, BL17]. Firstly, generate $a^{-1} * E, a * E, a^2 * E$ where $a^{-1} * E = O(a * E, E)$ and $a^2 * E = O(E, a * E)$. Next, given $(a^i * E, a^{i+1} * E)$, one can compute $a^{2i} * E$ and $a^{2i+2} * E$ by $O(E, a^i E)$ and $O(E, a^{i+1} E)$, resp. Besides, obtain $a^{2i+1} * E$ by $O(a^{-1} * E, a^i * E)$. Therefore, we can compute $a^n * E$ with Montgomery ladder method within $2\log(n)$ oracle queries. □

Proposition 2.1. *The square CSIDH problem is equivalent to the computational CSIDH problem if the order h of the group Cl is odd and given.*

Proof. Given the challenge $(E, a * E, b * E)$ and access to the oracle O. First, generate $a^2 * E, a^{-2} * E$ by $O(E, a * E), O(a^2 * E, E)$ iteratively. Then, generate $a^2 b^2 * E$ by $O(a^{-2} * E, b * E)$. By Lemma 2.1, get the element by computing $(\frac{h+1}{2})$-fold actions for $a^2 b^2 * E$ with respect to E.

We have $(a^2 b^2)^{\frac{h+1}{2}} * E = ab * E$, since $(a^2 b^2)^{\frac{h+1}{2}} = ab$. Note that, by the basic group theory, the square root of Cl of an odd order is unique through the isomorphism $\phi(x) = x^2$ whose kernel is trivial since $x^2 = y^2$ implies $x = y$. □

Remark. In the formal CSIDH setting ($p = 3 \mod 4$), the class number is odd [CLM+18]. To have a full reduction, more propositions of the ideal class group are exploited. A full version is provided in Appendix A for the remaining case. Therefore, we have the following relations.

Computational CSIDH $=_{quantum}$ **Computational Square CSIDH** $=_{classical}$ **Computational Inverse CSIDH**.

Problem 5. (Reciprocal CSIDH Problem) Given a curve E. Firstly, the adversary chooses and commits to $X \in \mathcal{E}$, then receives the challenge $s*E$ where $s \in Cl$. Then the adversary must compute a pair $(s*X, s^{-1}*X)$ with respect to the committed X.

Intuitively, the computational reciprocal CSIDH problem is a relaxed version of the square CSIDH problem or the inverse CSIDH problem. In particular, if one can solve the inverse CSIDH problem, then one can solve the reciprocal CSIDH problem by taking $X = E$ with $(s * X, s^{-1} * X) = (s * E, s^{-1} * E)$. Conversely, if an attacker knows the isogeny between X and E, or E^t, then this can be used to solve the *inverse CSIDH problem*. That is, if $X = r * E$, one can obtain $s^{-1} * E$ by computing $r^{-1} * (s^{-1} * X)$ with the given r. On the other hand, if $X = r * E^t$, one can obtain $s^{-1} * E$ by computing $r * (s * X)^t$ with the given r. However, note that the attacker is not required to know the isogeny between X and E or E^t in the problem.

The reciprocal CSIDH problem appears to be non-standard at first sight but, in fact, it's equivalent to the inverse CSIDH problem. Even though the problem concedes additional freedom X for the attacker, yet notice that X is chosen prior to the challenge $s * E$. The proposition leads to the result that uncertainty and freedom of X can be self-reconciled, as shown in the following reduction. We call the proposition as "self-reconciling".

Proposition 2.2. *The computational reciprocal CSIDH problem is equivalent to the computational inverse CSIDH problem.*

Proof. Given the challenge $(E, s * E)$ for the inverse CSIDH problem. Invoke the oracle for the reciprocal CSIDH with E. After receiving X from the oracle, send the challenge $s * E$ to the oracle. Receive $(s * X, s^{-1} * X)$ from the oracle, then rewind the oracle to the time when it output X, and then send $s * X$ as the challenge with respect to committed X again. Receive (X_0, X_1) from the oracle. Output X_1.

Claim $X_1 = s^{-1} * E$. Write $X = b * E$ by the transitivity of the action, so $s * X = (sb) * E$. Then, since the second challenge is $s * X = (sb) * E$, we have $X_1 = (sb)^{-1} * X = s^{-1} * E$. Precisely, if the adversary can solve the problem based on E with committed X with probability ϵ, then the adversary can be used to solve the inverse CSIDH problem based on E with probability ϵ^2.

□

In the proof Proposition 2.2, the reduction extracts the first entry of the first solution and the second entry of the second solution to obtain the solution for the inverse CSIDH problem. We can, therefore, conclude the following corollary.

Corollary 2.1. *In the experiment of Problem 5, after committing to the curve X, if the adversary can solve $(s * X, s'^{-1} * X)$ with respect to different given challenges $s * E$ and $s' * E$ then the adversary can be used to solve the computation inverse CSIDH problem.*

We end the subsection with the following relation for the CSIDH setting in [CLM+18] ($p = 3 \mod 4$). (A full reduction is provided in Appendix A.)

Computational CSIDH $=_{quantum}$ **Computational Inverse/Square/reciprocal CSIDH**.

Remark. The above results all extend to the discrete logarithm setting. We leave the details to the reader.

2.2 Functionalities

In this subsection, we define the functionalities we need as well as the related security definitions.

A symmetric encryption scheme is a pair of algorithms (E, D) defined over message space $\mathcal{M}$ and ciphertext space C with key space $\mathcal{K}$.

Definition 2.1. *(non-committing encryption (NCE)) A symmetric encryption scheme (E, D) is said to be non-committing if there exists PPT algorithm B_1, B_2 such that for any PPT distinguisher $\mathcal{D}$, message $m \in \mathcal{M}$.*

$$\left|\Pr[\mathcal{D}(c, k) = 1] - \Pr[\mathcal{D}(c', k') = 1]\right| = negl(n),$$

where $k \leftarrow_s \mathcal{K}, c = \mathsf{E}_k(m)$ and $c' \leftarrow_s B_1(1^n), k' \leftarrow_s B_2(c', m)$

Informally speaking, non-committing encryption allows a user to generate a dummy ciphertext indistinguishable from the real one by B_1 and later explain it with the assistance of B_2. The idea was introduced by Canetti et al. [CFGN96] with the one-time pad (OTP) as an instantiation. It was also used in some oblivious transfer constructions [CO15, BDD+17]. The non-committing proposition here is used to extract the input without rewinding in the simulation process. The proposition can be replaced with IND-CPA, but the simulation may thereby need rewinding.

$\mathcal{F}_{RO}$-Functionality of Random Oracle

The functionality is a function with the domain Cl and the codomain $\mathcal{K}$. It keeps a list L of pairs of the elements in Cl and $\mathcal{K}$ where the initial state is empty. It works as follows:

1. Upon receiving a query $C \in Cl$, check whether (C, k') for some $k' \in \mathcal{K}$. If so, set $k = k'$; if not, generate $k \leftarrow_{\$} \mathcal{K}$ and store the pair (C, k) in the list L.
2. Output k.

The functionality of a random oracle $\mathcal{F}_{RO}$ internally contains an initially empty list. Upon receiving the query from the domain, it will check whether it is a repetition. If so, return the value assigned before; otherwise, it randomly assigns a value from the codomain, stores the pair, and returns the value. Formally speaking, an input of a random oracle can be any binary string. On account of simplicity, we restrict the domain to Cl. This can be easily and compatibly extend to $\{0,1\}^*$, since the supersingularity can be efficiently verified [CLM+18].

$\mathcal{F}_{TSC}$-Functionality of a trusted setup curve

The functionality is to output an element of $\mathcal{E}$. It generates an ideal class $t \leftarrow_{\$} Cl$ and outputs the curve $t * E_0$.

The functionality of trusted setup curves $\mathcal{F}_{TRS}$ serves as a setup for generating a curve for the protocol. This setup hides the relation t between the public curve and the curve E_0. In practice, this can be replaced with a key exchange protocol [BF20]. That is, two parties do a key exchange first and obtain a curve that the isogeny relation to E_0 remains unknown.

Here we define the functionality of the oblivious transfer in simple and classic way. The two-party functionality of the oblivious transfer is characterized by $\mathcal{F}_{OT} = (f_1, f_2)$ where $f_1 : \{0,1\}^* \times \{0,1\}^* \to \{\perp\}$ and $f_2 : \{0,1\} \to \{0,1\}^*$. The functionality $\mathcal{F}_{OT} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\} \to \{\perp\} \times \{0,1\}^*$ takes in a message pair $x = (M_0, M_1)$ of equal length from one party and a bit $y = i \in \{0,1\}$ from the other party, and returns $\mathcal{F}_{OT}(x, y) = (f_1(x, y), f_2(x, y)) = (\perp, M_i)$ where $\perp$ represents an empty string.

We briefly define the security terms of OT. We refer [HL10, Lin17] for more details. Intuitively, we say a protocol realizes the functionality securely based on the simulation-based definition, if not only the protocol realizes the function but also whatever the adversary can learn from the real execution of the protocol can be indistinguishably generated by a simulator. Thus, we have to formalize the "view" of a corrupted party and compare the output of the protocol with the ideal functionality. Let π be a protocol computing $\mathcal{F}_{OT}$. The $view_i^{\pi}(x, y)$ represents the transcript that records whatever the i^{th} party sees during an execution of the protocol π taking input (x, y). Precisely, $view_i^{\pi}(x, y)$ is the tuple $(input, r^i, m_1^i, ..., m_n^i)$ where $input$ is the input of the party, r^i is its internal random tape, and m_j^i is the j^{th} received message. We also write $output_i^{\pi}(x, y)$ as the output received by the i^{th} party after the execution of the protocol π with the input (x, y), and write $output^{\pi}(x, y) = (output_1^{\pi}(x, y), output_2^{\pi}(x, y))$. In particular, if the protocol π completely realizes the function of the functionality $\mathcal{F}_{OT}$, then $output^{\pi}(x, y) = \mathcal{F}_{OT}(x, y)$.

Definition 2.2. *(security OT against semi-honest adversary) We say a protocol π securely (privately) computes $\mathcal{F}_{OT}$ in the presence of static semi-honest adversaries if there exists probabilistic polynomial-time algorithms S_1, S_2 such that*

$$output^{\pi}(x, y) = \mathcal{F}_{OT}(x, y)$$

$$\{S_1(x, f_1(x, y))\}_{x,y} =_c \{view_1^{\pi}(x, y)\}_{x,y}$$

and

$$\{S_2(y, f_2(x, y))\}_{x,y} =_c \{view_2^{\pi}(x, y)\}_{x,y}.$$

The notion implies that whatever the semi-honest adversary can learn from running the protocol, it could be generated by themselves without the execution. In other words, the semi-honest adversary can learn nothing more than allowed. The idea of ideal execution is implicit here. Since anything apart from the function of the functionality can be self-generated in an indistinguishable manner, the real protocol ideally realizes the functionality as long as the two parties follow the protocol specification (see Section 7.2 of [Ode09] for more details).

However, the semi-honest adversary model is indeed weak. It is inevitable in the real world that malicious users depart from the protocol specification with arbitrary strategies. A relaxation for the oblivious transfer protocol or single-output functionality is one-sided simulation. One-sided simulation, which requires the indistinguishability for the sender and the simulation for the receiver ensures privacy for both parties even in the presence of malicious adversaries, is also a plausible choice for an efficient construction. Here, we consider full-simulation in the presence of malicious adversaries.

Roughly speaking, the standard real/ideal paradigm demonstrates that for any kind of adversaries in the real world, there exists a corresponding simulator in the ideal world. This provides an ultimate guarantee for the construction that whatever the adversary can do in the real execution is inconsequential since they are predictable and simulatable in the ideal world. Hence, the definition of the real execution and the ideal execution should be clarified first.

Ideal Execution. The ideal execution captures a world where a trusted third party exists. The parties do not communicate with each other but instead hand their inputs to the trusted party. Then, the trusted party honestly returns the outcomes to each party, corresponding to the defined functionality. Nevertheless, the ideal execution in the presence of malicious adversaries is slightly different from the previous consideration of the semi-honest adversary. Due to losing the honest majority, fairness is not taken into consideration. Moreover, rational rebelling behaviors of the malicious adversaries, including refusing to participate, aborting the running sessions, or replacing the inputs, are taken into account. These strategies will be taken into account in the definition of the modified ideal functionality.

We define the modified ideal execution before going to the security definition. For more detailed exposition, also see [HL10, Lin17]. The ideal execution in consideration of the malicious adversary of a two-party functionality $\mathcal{F} = (f_1, f_2)$ consists of six phases: *initial inputs, inputs to the trusted party, early abortion, output to the adversary, instruction of continuing or halting, outputs*. Let P_i denote the corrupted party controlled by $\mathcal{S}$, P_j be the honest party where $\{i, j\} = \{1, 2\}$, $\mathcal{T}$ be for the trusted third party.

First of all, in the phase of *initial inputs*, like the ordinary setup, P_1 has the input x, P_2 has the input y and the adversary $\mathcal{S}$ has the auxiliary input z. Secondly, in the phase of *inputs to the trusted party*, honest P_j hands the initial input (x or y) to $\mathcal{T}$. What corrupted P_i sends is controlled by $\mathcal{S}$. The decision made by $\mathcal{S}$ including the early abortion option $\mathbf{abort}_i$ is based on the initial input of P_i and the auxiliary input z. Let (x', y') be the inputs to F. Thirdly, *early abortion* is an intermediate phase, if $\mathbf{abort}_i$ is sent within the second phase by $\mathcal{S}$. Then the trusted party returns $\mathbf{abort}_i$ to both parties, and the execution terminates; otherwise, the execution continues. Fourthly, in the phase of *output to the adversary*, $\mathcal{T}$ computes $f_1(x', y')$ and $f_2(x', y')$ and returns $f_i(x', y')$ to the corrupted party P_i first. Next, in the fifth phase, the adversary replies **continue** or $\mathbf{abort}_i$ to $\mathcal{T}$. This instructs $\mathcal{T}$ to continue or terminate by returning $f_j(x', y')$ or $\mathbf{abort}_i$ to P_j, respectively. Last but not least, in the final phase *outputs*, the honest party outputs $f_j(x', y')$. The adversary $\mathcal{S}$ in place of P_i outputs something based on the knowledge of the initial input (x or y), auxiliary input z, and $f_i(x', y')$.

The output pair of the honest party and the adversary from the ideal execution of the functionality $\mathcal{F}$ described above is denoted by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z),i}(x, y)$. Note that even though in oblivious transfer the sender receives no outputs from the trusted party, the adversary can still output something in place of the sender if the sender is the corrupted party.

Real Execution. The real execution is the execution of a real protocol. Let the protocol π compute the functionality $\mathcal{F}$ where the P_i is the corrupted party controlled by the adversary $\mathcal{A}$. The initial inputs are x for P_1, y for P_2 and the auxiliary input z for $\mathcal{A}$. During the execution of π, $\mathcal{A}$ will usurp P_i, interact with P_j, and finally output something. The messages and output provided by the adversary may deviate from the specification of π with a polynomial-time strategy. In contrast, the honest party P_j interacts with P_i and returns outputs as specified by the protocol. Let $\mathrm{REAL}_{\pi,\mathcal{A}(z),i}(x, y)$ denote the output pair by P_j and $\mathcal{A}$.

The aim of the standard real/ideal paradigm is to show that the ensemble produced by the simulator through the ideal execution is indistinguishable from the ensemble produced by the adversary via the real execution. This provides strong assurance of the security irrespective of the strategies the adversary adopts since any real adversary can be simulated in the ideal world. This also permits modular constructions for larger protocols by the composition theorems [Can00, Can01]. As a corollary, a relaxed but equivalent version of the security model is the simulation in the *hybrid model*.

Hybrid Model. The hybrid model contains real messages communicated between participants and oracle access to functionality $\mathcal{G}$ (ideal messages). The two-party protocol π with input (x, y) in a hybrid model with the functionality $\mathcal{G}$ is called the $\mathcal{G}$-hybrid model. In the presence of adversary $\mathcal{A}$ who controls the i^{th}

party with the auxiliary input z, we denote the output of all parties by $HYBRID^{\mathcal{G}}_{\pi,\mathcal{A}(z),i}(x,y)$.

We remark that this model is a prerequisite for constructing UC-secure oblivious transfer construction due to the impossibility results given in [CKL03]. In the $\mathcal{G}$-hybrid model, the simulator in the simulation process is able to exert control over the functionality $\mathcal{G}$ in simulation. For example, in the common reference string (CRS) hybrid model, two parties are given a shared string in the protocol execution, while in the simulation process, the simulator can invoke the adversary with a trapdoor string to cheat [PVW08].

To match the security definition presented in [Can01], assume there exists an environment machine $\mathcal{Z}$ serving as an interactive distinguisher between the real execution and the ideal execution. When interacting with the machine, the environment $\mathcal{Z}$ can decide all inputs of the parties and the auxiliary input for the adversary/simulator. After the execution, $\mathcal{Z}$ outputs a single bit to judge whether it interacts with a real machine or an ideal machine. Also, the environment $\mathcal{Z}$ can interact with the adversary/simulator with any queries at any time throughout the execution in order to distinguish. Here, we denote the ensemble consisting of the output of the ideal execution of the functionality $\mathcal{F}$ involving the adversary $\mathcal{S}$, the environment $\mathcal{Z}$ by $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and the ensemble consisting of the outputs in the hybrid model involving the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ by $HYBRID^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition 2.3. *(UC-realize) A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ in the presence of malicious adversaries and static corruption in the hybrid model with functionality $\mathcal{G}$ if for any adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for every interactive distinguisher environment $\mathcal{Z}$ we have*

$$IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}} \approx_c HYBRID^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}.$$

3 Our Proposal

This section introduces Chou and Orlandi's core of the OT scheme [CO15] to explain the idea of the tweaked key exchange. Secondly, we explain how we derive our novel compact proposal. Thirdly, we compress the three-round scheme to be round optimal (two-round) through the quadratic twist technique. Finally, based on the round-optimal structure, we add "the proof of ability to decrypt" rounds to extract the input of the receiver for completing the protocol.

3.1 Secure Schemes Against Passive Adversaries

Tweaked Key Exchange.

Figure 1 presents the Chou-Orlandi OT scheme [CO15], in a nutshell, which is based on the Diffie-Hellman key exchange. Both the sender and the receiver share their own public key g^s, g^r with each other. Both parties can secretly obtain a shared secret g^{rs}. In this OT protocol, the receiver uses this shared secret to decrypt the targeted message from the third flow. On the other hand, the receiver uses the second flow to obfuscate the secret bit (i).

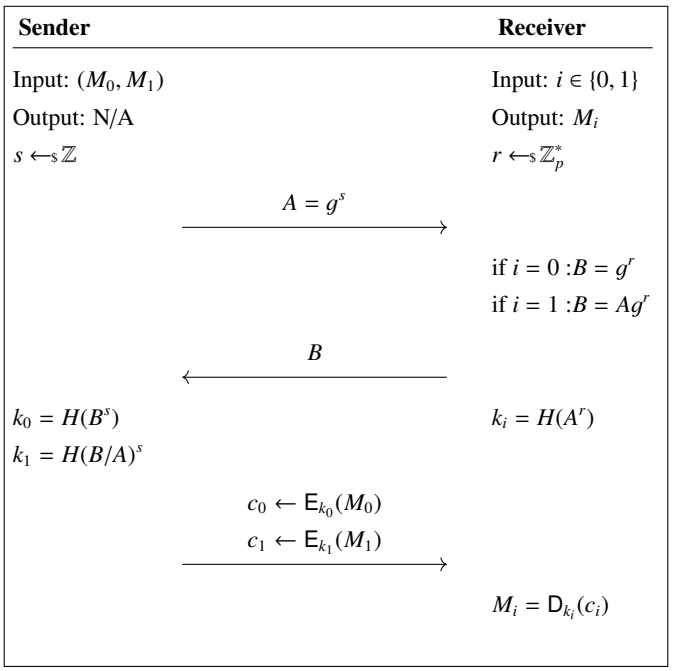

Figure 1: Chou and Orlandi's OT scheme in a nutshell [CO15]

We can interpret all aforementioned isogeny-based oblivious transfer constructions from this point of view. In [BOB18], the shared secret between the sender and the receiver is the j-invariant of the isomorphic elliptic curves $\phi_{B'}\phi_A(E)$ and $\phi_{A'}\phi_B(E)$. The receiver hides his intention by masking his $p_3^{e_3}$-torsion subgroup public basis by a pair of special $p_3^{e_3}$-torsion points $U, V \in \phi_B(E)$. Alice (the sender) thereby requires the same pair of points U, V to reconcile the noise. Hence, a coin-flipping mechanism is deployed in the protocol to guarantee that both the sender and the receiver can obtain the same points U, V. In order to prevent information leaks, there are three additional requirements for the pair (U, V) to generate, see [BOB18].

Other proposals [dSGOPS18, Vit19] also rely on the similar idea of using a fixed agreement key from the key exchange to decrypt the specific ciphertext. In the first OT construction in [dSGOPS18], it requires two public curves during the trusted setup, which serves as two agreement keys from the perspective of key exchange. In [Vit19], one more secret exponent/$p_2^{e_2}$-torsion subgroup generated by the sender is required to obtain two agreement keys.

Our 3-round protocols.

We overcome this with a different strategy. The key idea is that both the sender and the receiver can raise to power s, s^{-1} and r, r^{-1}, respectively. Instead of relying on the single Diffie-Hellman key g^{rs}, two key agreement schemes are used. Each of them relies on a different but equivalent hardness assumption. The receiver will possess two relevant secrets from a single public key. The main idea of our scheme is combining the equivalent key exchange schemes from Problem 1 and Problem 2 to obtain "partially dual agreement keys." Up to the receiver's secret bit, one serves as a formal decryption key (agreement key), and the other one will serve as the message for the sender.

In fact, without revealing, the receiver possesses two secrets g^r and g^{sr}. Only the receiver would distinguish them with the knowledge of r and g^s. Instead of using the shared secret g^{sr} to decrypt, the receiver decrypts messages with one of them. Specifically, he decrypts c_0 with g^{sr} if the secret bit is 0, and decrypts c_1 with g^r if the secret bit is 1. After receiving one of the two secrets, the sender encrypts one by doing one Problem 1 computation and one Problem 2 computation. Intuitively, the receiver's intention remains obscure since only one public key is exposed. The other message of the sender keeps secure due to the hard problem. To be more specific, the receiver has g^r, g^{rs} as the sender has $\{g^{rs}, g^{s^{-1}r}\}$ or $\{g^{rs^2}, g^r\}$. This corresponds to the inverse CSIDH problem or the square CSIDH problem, respectively. Based on this idea, we derive the prototype of our OT protocol as follows.

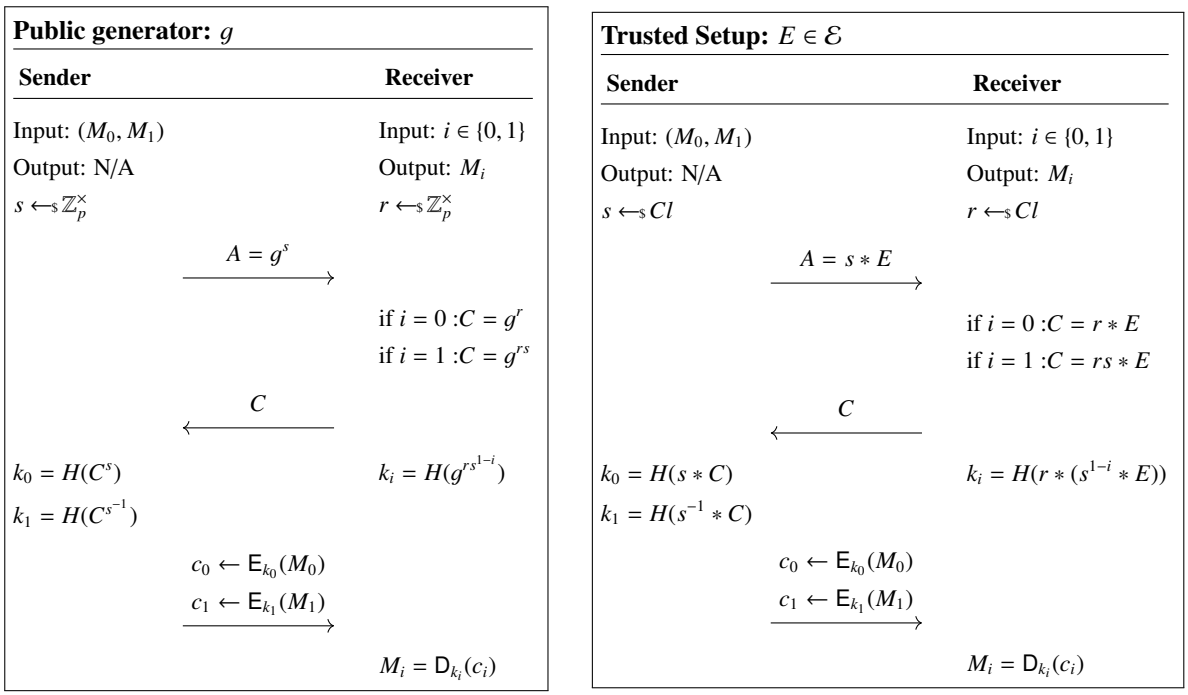

Figure 2: The prototypes of our 3-round OTs.

Note that a requirement for the isogeny-based protocol in Figure 2 is that the isogeny relation between public curve $E \in \mathcal{E}$ and E_0 should remain unknown. The reason is that if the receiver knows that $t * E_0 = E$, then the receiver can input $i = 0$ and compute the other key by $t^2r^2 * (rs * E)^t = t^2r^2 * (trs * E_0)^t = trs^{-1} * E_0 = rs^{-1} * E$.

As shown in Figure 2, the three-round protocol, however, is somewhat awkward. Notice that the secret bit of the receiver is committed in the second message after receiving the first message from the sender. It is imaginable that the receiver corrupted by the malicious adversary can send arbitrary $C \in \mathcal{E}$ (if we put a supersingularity verification process). Thus, the hardness assumption is the weaker version of the reciprocal CSIDH problem: given $(E, s * E)$ to find $(X, s * X, s^{-1} * X)$ where $X \in \mathcal{E}$. Consequently, this not only undermines the underlying hardness of the protocol but also makes it impossible to prove equivalence to the computational CSIDH problem.

In this protocol, we believe it is impossible to demonstrate the receiver's ability to decrypt without using a zero-knowledge proof of isogenous relation in a secure way. Since the receiver's message C and the decryption key k_i are bound together by a fixed isogeny s and the sender cannot let both keys exposed, no simple processes can achieve the desired functionality. The reason is that the corrupted sender can always replace keys with random strings to extract the receiver's secret bit. As a result, we should compress the protocol.

Our 2-Round Protocol.

Notice that in the protocol in Figure 2, the sender computes $s * C, s^{-1} * C$ as encryption keys where $s^{-1} * C = (s * C^t)^t$. Also, notice the quadratic twist of $r^{-1} * (s * E)^t$ is exactly $r * (s * E)$. Based on these two main ideas, we utilize the quadratic twists to compress our 3-round protocol into a 2-round protocol as shown in Figure 3.

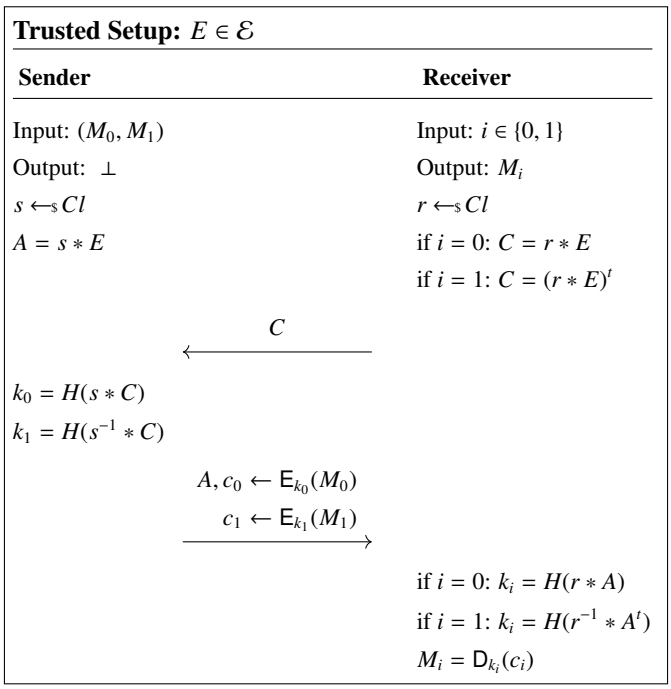

Figure 3: The core of our 2-round OT scheme. Note that no analogues exist in the Diffie-Hellman setting due to the quadratic twist.

We can also do an easy check for the correctness in the semi-honest model that if $i = 0$, then $s * C = s * (r * E) = r * A$; if $i = 1$, then $s^{-1} * C = s^{-1} * (r * E)^t = r^{-1} * (s^{-1} * E^t) = r^{-1} * (s * E)^t = r^{-1} * A^t$. As shown in Figure 3, the computations almost remain the same as the previous scheme. The number of isogeny computations is the same: 3 for the sender and 2 for the receiver. The quadratic twisting is an efficient operation via the field negation.

On the aspect of security, the intuition beneath the protocol is that the curious but honest sender cannot discern the intention of the receiver, since the distributions of the first messages are the same with respect to the input i. Also, the curious but honest receiver cannot read the unchosen message unless the receiver can know the corresponding key, which is the hash value of the answer to the inverse CSIDH problem. The formal simulation-based security notion, however, requires a more rigorous indistinguishable simulation process for the distinguisher, see Section 4.

Remark. To make the protocol succinct, we slightly modify the protocol by having the sender compute $s * C, s * C^t$, and the receiver compute $r * A$ as the key. One can verify that two versions are equivalent.

3.2 The Full Construction Against Malicious Adversaries

The full protocol is shown in Figure 4 below. The variant follows the remark above. To be secure against malicious adversaries who may deviate from the specification, both parties will do a simple verification of the received elements to avoid a variety of unpredictable outcomes. In the CSIDH setting, both parties will check whether the curve is supersingular, which can be done efficiently, as shown in [CLM+18].

In order to extract the receiver's input, the receiver must demonstrate the ability to decrypt. The reason to do this is that the corrupted receiver who skips all hash queries makes the input intractable to the simulator. Note that the mechanism should also ensure the security of the input of the receiver during the demonstration. Here the sender will send another curve $s' * E$ distinct from $s * E$ for transferring messages. The sender encrypts the ideal s' and a concatenated random string with key pair derived from $s' * E$. The receiver decrypts one ciphertext with X, and the other ciphertext serves as a verification of the equality of encrypted messages.

Protocol 1. (CSIDH-based OT) Let (E, D) be a symmetric encryption scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $H : \mathcal{E} \to \mathcal{K}$ be modeled as a random oracle $\mathcal{F}_{RO}$ that serves as the key derivation function from the group $\mathcal{E}$ to the key space $\mathcal{K}$ for the symmetric encryption scheme. We assume two parties have a probabilistic polynomial-time algorithm that verifies whether a curve is in $\mathcal{E}$.

- **Trusted Setup:** Let $E = t * E_0$ where $t \leftarrow_\$ Cl$ is unknown.

- **Input:**

  The sender $\mathcal{S}$ takes two messages M_0, M_1 as the input where M_0, M_1 have the same length. The receiver $\mathcal{R}$ takes a bit $i \in \{0, 1\}$ as the input.

- **Procedure:**

  1. $\mathcal{S}$ prepares ideals independently $s_0, s_1 \leftarrow_{\$} Cl$, a random string $str \in \{0,1\}^n$ and computes $A_0 = s_0 * E$, $A_1 = s_1 * E$.
  2. $\mathcal{R}$ generates an ideal $r \leftarrow_{\$} Cl$ then computes $C = r * E$. If $i = 1$, overwrite $C = C^t$. $\mathcal{R}$ sends C to $\mathcal{S}$.
  3. $\mathcal{S}$ checks whether $C \in \mathcal{E}$. If not, $\mathcal{S}$ aborts and outputs $\mathbf{abort}_2$. Otherwise, $\mathcal{S}$ computes four keys $k_{j,0} = H(s_j * C)$ and $k_{j,1} = H(s_j * C^t)$ for $j \in \{0, 1\}$. Then, $\mathcal{S}$ computes four ciphertexts $c_{0,j} \leftarrow \mathsf{E}_{k_{0,j}}(M_j)$ and $c_{1,j} \leftarrow \mathsf{E}_{k_{1,j}}(s_1 \parallel str)$ for $j \in \{0, 1\}$. $\mathcal{S}$ sends $(A_0, A_1, c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$ to $\mathcal{R}$.
  4. $\mathcal{R}$ runs the proof of ability to decrypt first. $\mathcal{R}$ checks whether $A_1 \in \mathcal{E}$. If not, $\mathcal{R}$ aborts and outputs $\mathbf{abort}_1$. Otherwise, $\mathcal{R}$ computes $k'_{1,i} = H(r * A_1)$ and $(s'_1 \parallel str') \leftarrow \mathsf{D}_{k'_{1,i}}(c_{1,i})$. Verify whether $s'_1 * (r * E) = r * A_1$. If not, output $\mathbf{abort}_1$. Otherwise, continue.
  5. $\mathcal{R}$ computes $k'_{1,1-i} = H(s'_1 * (r * E)^t)$. Verify whether $\mathsf{D}_{k'_{1,1-i}}(c_{1,i-i}) = (s'_1 \parallel str')$. If not, output $\mathbf{abort}_1$. Otherwise, continue.
  6. $\mathcal{R}$ verifies $A_0 \in \mathcal{E}$. If not, $\mathcal{R}$ aborts and outputs $\mathbf{abort}_1$. Otherwise, compute the decryption key $k'_{0,1} = H(r * A_0)$ and output $M_i \leftarrow \mathsf{D}_{k'_{0,i}}(c_{0,i})$. And send str' to $\mathcal{S}$.
  7. $\mathcal{S}$ checks whether $str = str'$. If not, $\mathcal{S}$ aborts and outputs $\mathbf{abort}_2$. Otherwise, $\mathcal{S}$ accepts and outputs $\perp$.

On the aspect of simulation-based security, the intuition behind the protocol is that the malicious adversaries controlling the sender cannot discern the intention of the receiver, since the distributions of the first messages are the same with respect to the input i. Further, the simulator sets up a trapdoor through controlling the functionality $\mathcal{F}_{TSC}$.

To simulate the malicious adversaries controlling the receiver, the simulator should extract the adversary's input by observing the hash queries. The additional "proof of ability to decrypt" mechanism forces the adversary either to abort or to prove the ability to decrypt. This makes the input of the adversary traceable to the simulator. Furthermore, since the simulator can only obtain one real message from the trusted third party (up to the extracted input i), the simulator must forge the other ciphertext by the non-committing encryption scheme. The difference between the unchosen ciphertexts is not noticeable unless the environment machine possesses the corresponding decryption key with which combining the other decryption key is exactly the solution for the reciprocal CSIDH problem. See Section 4 for more details.

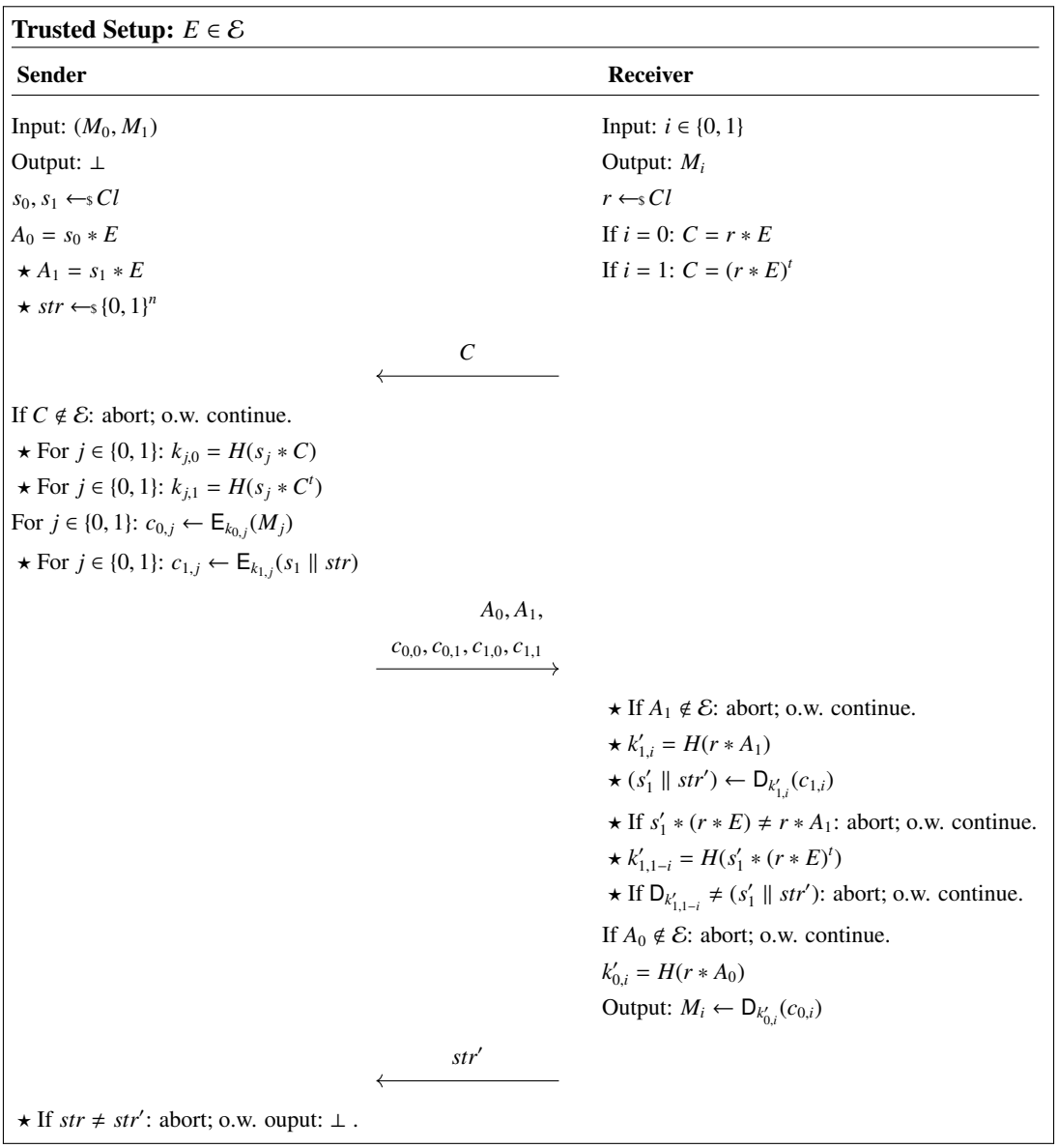

Figure 4: Our CSIDH-based oblivious transfer protocol. For the sake of readability, we label the steps related to the process of "proof of ability to decrypt" with $\star$.

Remark. The final round seems unnatural because the corrupted receiver can still get the message and skip the final round. In fact, it fits well with the ideal functionality since the adversary can also abort the session after receiving the value from the trusted party without outputting it.

4 Security Analysis

In this section, we consider three kinds of attackers and the underlying hard problems. We start with eavesdroppers and semi-honest adversaries as a warm-up. Then we consider the static malicious adversaries.

4.1 Eavesdropper and semi-honest adversary

Eavesdropper. An eavesdropper receives all the communications of parties and does not intervene in the execution. We will prove that the view of an eavesdropper of the execution of the 2-round protocol in Figure 3 can be simulated by a probabilistic polynomial time simulator without the inputs if the CSIDH problem is hard (Problem 1).

Note that we may assume the eavesdropper knows the parties' inputs while the simulator is given nothing. The reason for the assumption is to match the definition of UC-security [Can01] where the environment

machine decides the inputs.

Semi-Honest Adversary. A static semi-honest adversary who controls one party of the protocol from the beginning will follow the protocol specification and try to learn more information from the transcript. We will prove that the naive 2-round protocol in Figure 3 is secure against semi-honest adversaries if **the computational inverse CSIDH problem** or **the computational square CSIDH problem** is infeasible.

We remark that it is not meaningless to show the security against different but hierarchical adversaries. Since the adversaries are endowed with different abilities, the ability of the simulator varies accordingly. See [HL10] for an example where a protocol is secure against malicious adversaries but is not secure against semi-honest adversaries.

Theorem 4.1. *The protocol in Figure 3, denoted by π, where $H(.)$ is a random oracle and the encryption scheme (E, D) is IND-CPA, securely computes $\mathcal{F}_{OT}$ in the presence of static semi-honest adversaries if the computational inverse CSIDH problem (Problem 4) is infeasible. Moreover, the view of an eavesdropper of π can be indistinguishably simulated by a probabilistic polynomial time simulator without the inputs if the CSIDH problem is hard (Problem 1).*

Proof. (**Correctness**) Let $i \in \{0, 1\}$ be the input of the receiver $\mathcal{R}$. Say the sender $\mathcal{S}$ generates ideal $s \in Cl$ and $\mathcal{R}$ generates $r \in Cl$. If $i = 0$, then $C = r * E$. $\mathcal{S}$ computes the encryption key k_i as $H(s * C)$, and sends $A = s*E$. $\mathcal{R}$ computes $k'_i = H(r*A)$ as the decryption key. We have $r*A = r*(s*E) = s*(r*E) = s*C$. On the other hand, if $i = 0$, then $C = (r * E)^t$. $\mathcal{S}$ computes $k_i = H(s^{-1} * C)$ while $\mathcal{R}$ computes $k'_i = H(r^{-1} * A^t)$. We have $s^{-1} * C = s^{-1} * (r * E)^t = s^{-1} * (r^{-1} * E^t) = r^{-1} * (s^{-1} * E^t) = r^{-1} * A^t$. This shows the correctness of the protocol.

(**Corrupted sender $\mathcal{S}^*$**) The simulator S_1 takes as input $(M_0, M_1, \perp)$ and is required to simulate the view $view_1^{\pi}(M_0, M_1, i) = (M_0, M_1, rp, C)$ where rp is a random tape. S_1 performs these steps:

1. Uniformly generate a random tape rp for $\mathcal{S}^*$.
2. Generate $r' \leftarrow_{\$} Cl$.
3. Output $(M_0, M_1, rp, C' = r' * E)$.

On the other hand, $\mathcal{S}^*$ generates s with its own uniformly generated random tape and receives only one curve C from the honest receiver. The curve C is either $r * E$ or the quadratic twist $(r * E)^t$. Due to the sampling which is assumed to be indistinguishable from the uniform sampling, any distinguisher $\mathcal{D}$ with a given tuple (M_0, M_1, i) is not able to distinguish $\{S_1((M_0, M_1), \perp)\}_{(M_0,M_1),i}$ and $\{view_1^{\pi}(M_0, M_1, i)\}_{M_0,M_1,i}$.

(**Corrupted receiver $\mathcal{R}^*$**) The simulator S_2 takes as input (i, M_i) and is required to simulate the view $view_2^{\pi}(M_0, M_1, i) = (i, rp, A, c_0, c_1)$ where rp is a random tape. S_2 performs these steps:

1. Choose a uniform generated random tape rp for $\mathcal{R}^*$.
2. Generate $s' \leftarrow_{\$} Cl$, and generate $r' \leftarrow_{\$} Cl$ by the tape. Compute the curve C as $r' * E$ or $(r' * E)^t$ for the sender depending on i.
3. Compute the decryption keys k'_i, k'_{1-i} as an honest sender with s' and C. Replace k'_{1-i} with $k'' \leftarrow_{\$} \mathcal{K}$
4. Produce ciphertexts $c_i = \mathsf{E}_{k'_i}(M_i)$ and $c_{1-i} = \mathsf{E}_{k''}(garbage)$ where $garbage$ is a random generated string from the message space $\mathcal{M}$ of the same length as M_i.
5. Finally, the simulator outputs $(i, rp, s' * E)$ concatenated with (c_i, c_{1-i}) in the right order depending on i.

We claim that if there exists a PPT distinguisher which distinguishes between the simulated view and the real view with non-negligible advantage, then a reduction can be made to solve the computational problems (Problem 3 or the equivalent Problem 4) or to refute the IND-CPA of the encryption scheme.

To see this, a hybrid argument is used. Let $\mathcal{H}_0$ be the view of the real adversary, and $\mathcal{H}_2$ be the view generated by S_2 (i.e., $\{view_2^{\pi}(M_0, M_1, i)\}_{M_0,M_1,i}$ and $\{S_2((M_0, M_1), \perp)\}_{(M_0,M_1),i}$, resp). Let the intermediate $\mathcal{H}_1$ be the view of running the same protocol as $\mathcal{H}_0$ except that the encryption key k_{1-i} is replaced by a random one $k' \leftarrow_{\$} \mathcal{K}$. The main difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is that the real message M_{1-i} is replaced with a garbage string.

Hybrid 1. Claim $\mathcal{H}_0 =_c \mathcal{H}_1$ if the computational inverse CSIDH problem (Problem 4) is hard. Let $\mathcal{D}$ be the distinguisher for the experiment, then a solver for Problem 4 with the assistance of $\mathcal{D}$ is constructed as follows:

Input the challenge $(E', s' * E')$ where $s' \in Cl$ is unknown.

1. Set E' to be the public curve as in the protocol π and $s' * E'$ as the received curve for the receiver. (The reduction is to Problem 3 if $(E', s' * E')$ is reversed here.)
2. Randomly generate a random tape, generate r' by the tape, and act as the honest receiver.
3. While running, also simulate the random oracle in a standard way by assigning a random value from $\mathcal{K}$ whenever a query is made and record the queries during the execution.
4. Play the protocol π alone with self-generated input (M'_0, M'_1, i) as the specification except for the encryption part of the sender. In the encryption part, replace the encryption key k_{1-i} with $k' \leftarrow_{\$} \mathcal{K}$ to simulate the view of $\mathcal{H}_1$.
5. Invoke the distinguisher $\mathcal{D}$ with the self-generated view of $\mathcal{H}_1$.
6. If $\mathcal{D}$ judges as $\mathcal{H}_0$, restart with the pair $(E', s' * E')$ replaced by $(t * E', t's't * E')$ for two random isogenies $t, t' \leftarrow_{\$} Cl$. Otherwise, continue.
7. Randomly output a curve in the past queries of the simulated random oracle with a slight modification (see below). If it had been refreshed in Step 6 before, pull back with the isogeny $t't^{-1}$ first.

Note that the difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is the key for M_{i-1}, and both keys are generated uniformly from the key space $\mathcal{K}$. That is, a distinguisher for $\mathcal{H}_0$ and $\mathcal{H}_1$ with non-negligible advantage should know the corresponding curve for k_{1-i}. Precisely, let $\mathbf{E}$ denote the event that the targeted curve (specifically, $s'^{-1} * E$) is in the list, then we have

$$\begin{aligned}
&| \Pr[\mathcal{D}(\mathcal{H}_0) = 1] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1] | \\
=\ &| \Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \mathbf{E}] \times \Pr[\mathbf{E}] + \Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] \times \Pr[\neg\mathbf{E}] \\
&- \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \mathbf{E}] \times \Pr[\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}] \times \Pr[\neg\mathbf{E}] | \\
\leq\ &\Pr[E] + |\Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|,
\end{aligned}$$

where $|\Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|$ is negligible. As a result, if the distinguisher succeeds with non-negligible advantage, then the targeted curve is in the list with non-negligible probability.

After $\mathcal{D}$ succeeds, randomly output a curve from the list with the abovementioned processing as the solution. Let the reduction requires q_H queries of the random oracle to achieve f distinguishing advantage, then the reduction can solve one of Problem 3 or Problem 4 with probability greater than $(f - negl)/q_H$.

Hybrid 2. On the other hand, claim $\mathcal{H}_1 =_c \mathcal{H}_2$ for any PPT distinguisher if the encryption scheme (E, D) is IND-CPA. The only difference is the encryption $\mathsf{E}_{k'}(M_{1-i})$ in $\mathcal{H}_1$ and the encryption $\mathsf{E}_{k'}(garbage)$ in $\mathcal{H}_2$ where k' is uniformly sampled from $\mathcal{K}$. This shall hold due to the IND-CPA security of the encryption scheme. A reduction can be constructed through the following steps:

1. Run the protocol π with self-generated (M_0, M_1, i) except keeping the content of the variable c_{1-i} void.
2. Submit M_{1-i} and *garbage* to the IND-CPA challenger with an unknown encryption key.
3. Receive and assign the challenge ciphertext to c_{1-i}.
4. Invoke the distinguisher and input the generated view. Note that the distinguisher also knows (M_0, M_1, i) from the ensemble index.
5. Output the answer of the distinguisher.

The choice of $\{M_{1-i}, garbage\}$ made by the challenger in step 3 represents $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. That is, if there is a PPT distinguisher $\mathcal{D}$ for $\mathcal{H}_1, \mathcal{H}_2$, then the reduction can win the IND-CPA experiment with the same advantage.

(**Honest sender and honest receiver**) Assume that there is an eavesdropper who can see all communication between the two parties. We claim that there exists a probabilistic polynomial time simulator that can generate a tuple which is indistinguishable from the view of the eavesdropper.

Let $x = (M_0, M_1)$ and $y = i \in \{0, 1\}$ be the inputs for the sender and the receiver, respectively. Write the view of the eavesdropper as the tuple $(C, s * E, c_0 = \mathsf{E}_{H(s*C)}(M_0), c_1 = \mathsf{E}_{H(s^{-1}*C)}(M_1))$ where $r, s \leftarrow_{\$} Cl$ $C = r * E$ or $(r * E)^t$ as in the protocol specification. The simulator generates $r', s' \leftarrow_{\$} Cl$ and $m_0, m_1 \leftarrow_{\$} \mathcal{M}$, and outputs the simulated view $(C', s' * E, c_0 = \mathsf{E}_{H(s'*C)}(m_0), c_1 = \mathsf{E}_{H(s'^{-1}*C)}(m_1))$ where $C' = r' * E$.

Since we assume the sampling of Cl is indistinguishable from the uniform sampling, the distribution of the first message (i.e., C) is independent of the input of the sender i. The second messages of the eavesdropper and the simulator follow the same distribution. The remaining differences are encrypted messages since the eavesdropper does not know the isogeny class r, s in the real execution.

A hybrid argument is applied to show the indistinguishability again. The hybrid argument here is similar to the previous one. Let $\mathcal{H}_0$ be the view of the eavesdropper, and $\mathcal{H}_2$ be the tuple generated by the simulator. Insert with an intermediate view $\mathcal{H}_1$ which is the same as $\mathcal{H}_0$ except that both encryption keys are replaced with $k'_0, k'_1 \leftarrow_{\$} \mathcal{K}$.

Claim that $\mathcal{H}_0 = \mathcal{H}_1$ if the computational CSIDH problem (Problem 1) and its variant (Problem 2) are hard. Let $\mathcal{D}$ be the distinguisher between $\mathcal{H}_0$ and $\mathcal{H}_1$. Given the challenge $(E', a * E', b * E')$ of the problems, a concrete reduction can be made as follows:

1. Generate $M'_0, M'_1 \leftarrow_{\$} \mathcal{M}$.
2. Generate $t_0, t_1, t_2 \leftarrow_{\$} Cl$ for randomization.
3. Setup the public curve for the protocol to be $t_0 * E'$.
4. While running, also simulate the random oracle in a standard way by assigning a random value from $\mathcal{K}$ whenever a query is made and record the queries during the execution.
5. Give the input $((t_1t_0) * E', (t_2t_0) * E', \mathsf{E}_{k_0}(M'_0), \mathsf{E}_{k_1}(M'_1))$ to the distinguisher where $k'_0, k'_1 \leftarrow_{\$} \mathcal{K}$.
6. If the distinguisher outputs 1, then randomly pick a curve W in the hash list, and output $(t_0t_1t_2)^{-1} * W$.

The probabilistic analysis is the same as the previous one. Since the encryption keys in $\mathcal{H}_0$ and $\mathcal{H}_1$ are all uniformly generated from the key space, if a distinguisher can notice the difference, then the distinguisher should know the corresponding curves for queries.

Precisely, let $\mathbf{E}$ denote the event that the targeted curve is in the list, then we have

$$|\Pr[\mathcal{D}(\mathcal{H}_0) = 1] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1]| \leq \Pr[E] + |\Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|,$$

where $|\Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|$ is negligible. As a result, if the distinguisher succeeds with non-negligible advantage, then the targeted curve is in the list with non-negligible probability.

We also have $\mathcal{H}_1 = \mathcal{H}_2$ if the encryption scheme is IND-CPA. We omit the proof here since the proof is similar to the previous one in the case of the corrupted receiver.

Hence, we have $\{S_2(i, f_2((M_0, M_1), i))\}_{M_0,M_1,i} =_c \{view_2^{\pi}(M_0, M_1), i)\}$ and therefore the protocol π securely computes the functionality $\mathcal{F}_{OT}$ in the presence of a semi-honest adversary.

□

4.2 Malicious Adversary

Malicious Adversary. Next, we will prove the full construction in Figure 4 UC-realize the functionality $\mathcal{F}_{OT}$ in the presence of malicious adversaries. We consider static corruptions and malicious adversaries in charge of specific parties who are chosen before the execution and will deviate the protocol specification.

Theorem 4.2. *The protocol 1 of Section 3.2, denoted by π, where the encryption scheme (E, D) is non-committing, securely UC-realizes the functionality $\mathcal{F}_{OT}$ in the hybrid model with the functionality $\mathcal{F}_{RO}$ and a trusted setup $\mathcal{F}_{TSC}$ in the presence of malicious adversaries and static corruption if the computational reciprocal CSIDH problem is infeasible.*

Proof. **(Honest Sender and Honest Receiver)** We start with the honest sender and the honest receiver. The goal is to show that the execution of π is indistinguishable from the ideal functionality as the parties follow the specification. A similar proof as in Theorem 4.1 is sufficient since the view-based proof is equivalent to the simulation-based proof in the semi-honest (or honest) model, as stated in Section 2. The additional round does not reduce the security since the ideals s_0, s_1 are independently generated, the sender can self-generate a random string and the receiver can self-generate s_1 and a random string to recreate the view.

(Corrupted Sender and Corrupted Receiver) When two parties are corrupted, the simulator can invoke $\mathcal{A}$ with the input $(x = (M_0, M_1), y = i, z)$ given by the environment $\mathcal{Z}$, then the simulator outputs whatever the adversary outputs for both parties to complete the simulation.

(Honest Sender and Corrupted Receiver) Let $\mathcal{A}$ be the malicious adversary controlling the receiver. In order to emulate the adversary, it's necessary that the simulator extracts the input of the adversary and then sends it to the trusted party in the ideal execution. Say the environment $\mathcal{Z}$ generates input $(x = (M_0, M_1), y = i, z)$ and gives (y, z) to the simulator. The simulator S_2 passes any query from $\mathcal{Z}$ to $\mathcal{A}$ and returns the output of $\mathcal{A}$. The simulator S_2 with auxiliary input (y, z) proceeds the protocol execution with the adversary as follows:

1. Firstly, the simulator S_2 emulates a random oracle $\mathcal{F}_{RO}$ by keeping a list L of domain $\mathcal{E} \times \mathcal{K}$ and further records every past query. It initializes the random oracle with an empty list L. If the simulator receives a query of $E' \in \mathcal{E}$, the simulator checks whether $(E', k') \in L$ for some $k' \in \mathcal{K}$. If not, generate $k' \leftarrow_\$ \mathcal{K}$ and add the entry (E', k') to the list L. Finally, S_2 returns k' to emulate the random oracle.

2. Invoke the adversary $\mathcal{A}$ with the input (y, z). Generate the public curve $E = t * E_0$ by sampling $t \leftarrow_\$ Cl$ to simulate $\mathcal{F}_{TSC}$. Receive a curve X from the adversary.

3. Check whether $X \in \mathcal{E}$, if not, end the session by outputting $\mathbf{abort}_2$ to the functionality. Otherwise, continue.

4. Activate the algorithm B_1 of the non-committing encryption scheme. Generate $c_{0,0}, c_{0,1}$ with B_1, $s_0, s_1 \leftarrow_\$ Cl$ and $str \leftarrow_\$ \{0,1\}^n$. Compute A_0, A_1 and $c_{1,0}, c_{1,1}$ as the honest sender. Send $(A_0, A_1, c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$ to the receiver.

5. Process the external hash queries made as follows. Regarding the hash queries of curves $E' \in \mathcal{E}$, check whether $E' = s_j * X$ or $s_j * X^t$ for $j \in \{0,1\}$. If not, process it in a standard way. Otherwise, check further whether both $s_0 * X$ and $s_0 * X^t$ (i.e., the other decryption key) have been queried. If so, abort the session by outputting $\mathbf{abort}_2$. Otherwise, if E' is listed in the past queries $(E', k') \in L$, then return k'. Otherwise, send the ideal message i to $\mathcal{F}_{OT}$ where $i = 0$ for the case $E' = s_j * X$ or $i = 1$ for the case $E' = s_j * X^t$. After obtaining M_i from $\mathcal{F}_{OT}$, generate $k' \leftarrow B_2(c_{0,i}, M_i)$ and store $(s_0 * X, k')$ in the list. For the remaining cases ($j = 1$), process the hash query in a standard way.

6. After receiving str' from the adversary as the third flow of the protocol π, verify $str = str'$. If not, end the session by outputting $\mathbf{abort}_2$. Otherwise, continue.

7. After the outputs of the adversary, if $s_j * X$ or $s_j * X^t$ for $j \in \{0,1\}$ are not in the pair of L as the first entry, then end up the session with outputting $\mathbf{abort}_2$. Otherwise, the simulator output whatever the adversary outputs.

We claim $\{HYBRID^{\mathcal{F}_{RO}, \mathcal{F}_{TSC}}_{\pi, \mathcal{A}(z), 2}(x, y)\}_{x,y,z} =_c \{IDEAL_{\mathcal{F}_{OT}, S_2(z), 2}(x, y)\}_{x,y,z}$. We analyze the indistinguishability by comparing to the real execution. In the comparison, the abort in Step 5 implies the solution to the reciprocal CSIDH problem $(E, s * E)$ lying in the list L, which contradicts the assumption. Another abort is made in Step 7 after the pass in Step 6. The abort implies the adversary decrypts the ciphertext $c_{1,j}$ without the knowledge of the key. If this occurs with non-negligible probability, then it contradicts the non-committing assumption since the real ciphertext, the only thing contains information, can be decrypted without the key, while the dummy ciphertext cannot be (because it can be generated before the plaintext by B_1).

Other differences caused by the simulator are the ciphertexts for the receiver. Among the pair $(c_{0,0}, c_{0,1})$, thanks to the non-committing encryption scheme, $c_{0,i}$ is indistinguishable from the one in the real execution. The only suspicious part is $c_{0,i-1}$, which is a dummy ciphertext generated by the algorithm B_1 of encryption scheme. The counterpart in the real execution is $\mathsf{E}_{k_{1-i}}(M_{1-i})$ where k_{1-i} is either $H(s * X)$ or $H(s^{-1} * X)$.

Similar to the previous proof, the distinguisher (the environment machine) cannot succeed with non-negligible advantage only with the knowledge of k_{1-i}. Precisely, let $\mathbf{E}$ denote the event that the targeted curves $s * X, s * X^t$ are both queried where $(s * X^t)^t = s^{-1} * X$. We have

$$|\Pr[\mathcal{Z}(\mathcal{H}_0) = 1] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1]| \leq \Pr[E] + |\Pr[\mathcal{Z}(\mathcal{H}_0) = 1 \mid \neg E] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1 \mid \neg E]|\,.$$

Claim that the term $|\Pr[\mathcal{Z}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|$ is negligible if the encryption is non-committing. Given the non-committing challenge, denoted by (c, k). A concrete reduction can be made as follows:

1. Randomly generate $j \in \{0, 1\}$.
2. Run as the simulator with the environment machine except that assign value c to the variable $c_{0,j}$ in Step 4.
3. If the simulation process extracts i, the input of the receiver, is not j, then abort and restart the session.
4. If the environment machine judges the machine as the ideal machine, then output 1. Otherwise, output 0.

If the environment can win with non-negligible advantage $p(n)$ without the knowledge of the key, then the reduction can win the non-committing challenge with non-negligible advantage $p(n)/2$ where the loss is caused by the guessing in Step 2.

Since $|\Pr[\mathcal{D}(\mathcal{H}_0) = 1 \mid \neg\mathbf{E}] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1 \mid \neg\mathbf{E}]|$ is negligible, we have $|\Pr[\mathcal{D}(\mathcal{H}_0) = 1] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1]| \leq \Pr[\mathbf{E}] + negl(n)$. If the distinguisher can succeed with non-negligible advantage, then the solution of the reciprocal CSIDH problem (Problem 5) is in the list of the hash queries with non-negligible probability. Let the challenge of the reciprocal CSIDH problem start with E. A concrete reduction can be made as follows:

1. Run as the simulator with the environment machine, commit to the curve X obtained in Step 3 in the reciprocal CSIDH experiment and receive $s * E$.
2. Run as the simulator with the environment machine except that in Step 4 assign $s * E$ to the variable A_0.
3. Run as the simulator with the environment machine except that in Step 5, randomly pick a curve X_1 of $\mathcal{F}_{RO}$ query, and assign it with B_2 to the decryption key k_i since s is unknown. If the input bit i has not been extract, then randomly guess $i \in \{0, 1\}$.
4. If the environment machine judges the machine as the ideal machine, then randomly pick a curve X_2 in the hash query list as the first entry, and output (X_1, X_2). Otherwise, restart the challenge.

If the environment can win with non-negligible advantage $p(n)$ with q hash queries, then the reduction can win the non-committing challenge with non-negligible advantage $p(n)/(2q^2)$ where the loss is caused by the guessing in Step 3 and 4. To sum up, the simulator indistinguishably simulates the adversary if the encryption scheme is non-committing, and the reciprocal CSIDH problem is infeasible.

We remark that the simulator correctly extracts the input of the adversary in Step 5. According to Corollary 2.1, if the simulator extracts the wrong input, then the adversary can also be used to solve the inverse CSIDH problem.

(Corrupted Sender and Honest Receiver) Let $\mathcal{A}$ be a malicious adversary controlling the sender. In order to emulate the adversary, it's necessary for the simulator to extract the input of the adversary and to send it to $\mathcal{F}_{OT}$ in the ideal execution. The input here is exactly the message an honest receiver will read. Say the environment $\mathcal{Z}$ generates input $(x = (M_0, M_1), y = i, z)$ and gives (x, z) to the simulator. The simulator S_1 with input (x, z) proceeds as follows:

1. Firstly, the simulator S_2 emulates a random oracle $\mathcal{F}_{RO}$ by keeping a list L of domain $\mathcal{E}\times\mathcal{K}$ and further secretly records every past query. It initializes the random oracle with an empty list L. Whenever it receives a query of $E' \in \mathcal{E}$, the simulator checks whether $(E', k') \in L$ for some $k' \in \mathcal{K}$. If not, it generates $k' \leftarrow_{\$} \mathcal{K}$ and adds the entry (E', k') to the list L. Finally, S_2 returns k' to emulate the random oracle.
2. To simulate $\mathcal{F}_{TSC}$, generate $t \leftarrow_{\$} Cl$ and setup the public curve $E = t * E_0$ for the protocol π with the trapdoor t. Invoke the adversary $\mathcal{A}$ with input (x, z) and activate the session with the public curve E.

3. Generate $r \leftarrow_{\$} Cl$ and compute $C = r * E$. Send C to the adversary and act as the procedure of an honest receiver with the input $i = 0$ throughout the execution. (Note that the simulator does not know the input of the receiver here.)

4. If the adversary aborts, then send $\textbf{abort}_1$ to $\mathcal{F}_{OT}$ and finish the session. Otherwise, assume the execution is not aborted. Say receive $(A_0, A_1, c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$ from the adversary. Compute $k_0 = H(r * A_0)$, $k_1 = H((tr * (t^{-1} * A_0)^t)^t)$, and $m_j = \mathsf{D}_{k_j}(c_{0,j})$ for $j \in \{0, 1\}$.

5. Send $\{m_0, m_1\}$ to the trusted third party ($\mathcal{F}_{OT}$), output whatever the adversary outputs to complete the simulation.

Claim $\{HYBRID^{\mathcal{F}_{RO},\mathcal{F}_{TSC}}_{\pi,\mathcal{A}(z),1}(x, y)\}_{x,y,z} =_c \{IDEAL_{\mathcal{F}_{OT},\mathcal{S}(z),1}(x, y)\}_{x,y,z}$. In contrast to the real execution, there are two differences here. Firstly, the simulator possesses the trapdoor of the public curve. The process is identical to $\mathcal{F}_{TSC}$, and the simulator acts as an honest receiver throughout the process. Hence, the difference is unnoticeable to the adversary. Meanwhile, since the distribution of C in the protocol in the case of $i = 0$ is identical to the process of $i = 1$, it suffices to show the correctness of the extraction.

For the honest receiver who sends C to the sender with the input $i = 1$, then the private ideal is equivalent to $r^{-1}t^{-2}$ since $(r^{-1}t^{-2} * E)^t = (r^{-1}t^{-1} * E_0)^t = (r^{-1} * E^t)^t = (r * E) = C$. Hence, the receiver in the real execution decrypts $c_{0,1}$ with the curve $r^{-1}t^{-2} * A_0$. As our prediction of the key $k_1 = H((tr * (t^{-1} * A_0)^t)^t) = H((tr)^{-1}t^{-1} * A_0) = H(r^{-1}t^{-2} * A_0)$, the receiver will therefore get $m_1 = \mathsf{D}_{k_1}(c_{0,1})$. Hence, the simulator perfectly simulates in the ideal execution. Finally, We have $IDEAL_{\mathcal{F}_{OT},\mathcal{S},\mathcal{Z}} =_c HYBRID^{\mathcal{F}_{RO},\mathcal{F}_{TSC}}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Remark. In the formal description of [Can01], the environment machine and the adversary (simulator) starts with z, and the inputs of the parties are given through further instruction messages. Regarding readability and simplicity, we combine them into a single statement here without undermining the effectiveness of the proof.

5 Comparison

Table 1 gives a comparison between our oblivious transfer protocols with others [BOB18, dSGOPS18, Vit19] in terms of efficiency, including the number of curves in domain parameters, the number of curves in public keys for the sender and the receiver, the number of isogeny computations for the sender and the receiver, and the number of rounds, respectively. Among the isogeny-based OTs, our 2-round OT proposal is the most efficient on each aspect against semi-honest adversaries. It only takes additional one round and two isogeny computations for both parties to achieve UC-secure against static malicious adversaries.

Proposal	DP	PK_S	PK_R	$\#\ Iso_S$	$\#\ Iso_R$	$\#\ rounds$	Others
Barreto et al. [BOB18]	1	1	1	3	2	3	SIDH-based
Guilhem et al. I [dSGOPS18]	2	1	1	3	2	2	
Guilhem et al. II [dSGOPS18]	1	3	1	5	2	3	
Vitse [Vit19]	1	2	1	4	2	3	Insecure in CSIDH setting
This paper (Figure 3)	1	1	1	3	2	2	CSIDH-based
This paper (Figure 4)	1	1	1	5	4	3	CSIDH-based

Table 1: Comparison between isogeny-based OTs on efficiency. We separate the costs of 2-round core in Figure 3 and the full construction in Figure 4.

In [BOB18], the property of SIDH is used. The receiver randomly subtracts two selected points $U, V \in E_B$ to the points $(\phi_B(P_A), \phi_B(Q_A))$ to produce public points $(\hat{G}_A, \hat{H}_A)$ with respect to the secret bit i. The sender adds the same points jU, jV to the received points for $j \in \{0, 1\}$ to produce two decryption keys. The additional mechanism allows the receiver and the sender to generated the same points U, V. As stated in their work, randomly generated $U, V \in E_B$ may reveal the secret bit to the honest-but-curious sender by checking the equality of Weil pairings $e(P_A, Q_A)^{\ell_A}$, $e(\hat{G}_A, \hat{H}_A)$, and $(\hat{G}_A + \lambda U, \hat{H}_A + \lambda V)$ for $\lambda \in \mathbb{Z}$. On the other hand, it is also possible that the honest-but-curious receiver gets the isomorphic curves. In order to prevent these, the U, V are generated through a delicate process.

In [dSGOPS18], they proposed a secure UC-realize framework of oblivious transfer protocols against semi-honest adversaries. The DH, SIDH, CSIDH settings can fit into the framework. The first construction is a two-message oblivious transfer. It requires one more curve in the trusted setup phase.

In [Vit19], they proposed a construction based on the idea of exponentiation-only Diffie-Hellman. The construction can fit in the DH, SIDH, CISDH settings. But, as stated in their work, it will be totally insecure in the CSIDH setting against a malicious receiver. Specifically, their two-inverse problem is given curves $(E, a*E, b*E)$ to find some curve tuple $(X, a^{-1}*X, b^{-1}*X)$ where X is isogenous to E. This can be done in CSIDH setting by taking quadratic twist curves of $(E, a*E, b*E)$.

	Adversary Model	Security Definition	Model
Barreto et al. [BOB18]	Semi-honest*	Simulatable*	ROM+CRS
Guilhem et al. I [dSGOPS18]	Semi-honest	UC-realize	ROM+TSC
Vitse [Vit19]	Malicious	Semantic	Plain
This paper (Figure 3)	Semi-honest	UC-realize	ROM+TSC
This paper (Figure 4)	Malicious	UC-realize	ROM+TSC

Table 2: Comparison between others' isogeny OTs and our construction in Figure 4. The models includes the random oracle model (ROM), the common reference string model (CRS) and trusted setup curves (TSC). In [BOB18], a claim that the protocol ensures privacy in the malicious model is incorrect. The adapted definition is definition 2.6.1 of [HL10] that, as the statement, guarantees the privacy in the presence of *malicious* adversaries for a *2-rounds* oblivious transfer protocol while the scheme in [BOB18] has 3 rounds. For trusted setup curves (TSC), we believe it can be replaced by doing a key exchange first as [BF20] to achieve the same effect.

On the issue of security, a comparison is shown in Table 2. In [BOB18], the claim is incorrect. Except for the misuse of the definition, only limited malicious strategies are taken into account in the analysis. For instance, there could be potential malicious strategies for the receiver since the sender yields special points U, V from the points given by the receiver. Besides, the restrictions of U, V only prevent the information leakage for the passive case. Nevertheless, the proof at least demonstrates the simulatablility against semi-honest adversaries.

In [dSGOPS18], the protocols are universally composable secure in the semi-honest model. In both constructions, they also require trusted setup curves, as shown in Table 1. In [Vit19], they proposed a security definition called the semantic security of the oblivious transfer, which guarantees the indistinguishability for the sender within the distinct executions. However, the scheme is under a weaker decisional problem, and the security notion ensures nothing as coordinated in a larger construction.

6 Conclusion

In this paper, we present the first UC-secure isogeny-based oblivious transfer protocol in the presence of static corruptions and malicious adversaries. The result not only shows the existence of the construction but also presents a practical proposal. Despite the lack of freedom that DH-based oblivious transfer has over a field $\mathbb{F}_p$, the construction remains simple and compact, and the number of isogeny computations is constant. Moreover, the scheme shares the same hardness as the CSIDH key agreement scheme.

To achieve this outcome, we develop six techniques in this work. In the beginning, we mix the key-exchange-type problem and an equivalent variant to reduce the communication bandwidth. By utilizing the quadratic twists, we not only compress the number of rounds of the protocol but also fortify the hardness of the underlying assumption. Also, by combining the self-reconciling proposition and proof of ability to decrypt at the cost of one extra round, we are able to extract the input of the receiver to achieve one-side simulation. Furthermore, for the purpose of extracting the input of the sender, we setup trapdoors for the protocol via a new use of the quadratic twists to get a fully-simulatable construction. Finally, we develop a new computational assumption and prove it is equivalent to the standard CSIDH assumption with a quantum reduction.

We remark that these techniques are not exclusive to isogeny-based cryptography except for the quadratic twists. We envisage that these techniques can serve as potential cryptographic tools in future work.

Acknowledgments

We thank Wouter Castryck for his comments. This research is funded by the Ministry for Business, Innovation and Employment in New Zealand.

This work has been also supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ERC.

References

[BDD+17] Paulo SLM Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson CA Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. *arXiv preprint arXiv:1710.08256*, 2017.

[BdFLS20] Daniel Bernstein, Luca de Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *arXiv preprint arXiv:2003.10118*, 2020.

[BF20] Jeffrey Burdges and Luca De Feo. Delay encryption. Cryptology ePrint Archive, Report 2020/638, 2020. `https://eprint.iacr.org/2020/638`.

[BFJ16] Jean-François Biasse, Claus Fieker, and Michael J Jacobson. Fast heuristic algorithms for computing relations in the class group of a quadratic order, with applications to isogeny evaluation. *LMS Journal of Computation and Mathematics*, 19(A):371–390, 2016.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: Efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019*, pages 227–247, 2019.

[BL17] Daniel J Bernstein and Tanja Lange. Montgomery curves and the montgomery ladder. *IACR Cryptology ePrint Archive*, 2017:293, 2017.

[BM89] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and spplications. In *CRYPTO '89*, pages 547–557, 1989.

[BOB18] Paulo Barreto, Glaucio Oliveira, and Waldyr Benits. Supersingular isogeny oblivious transfer. *arXiv preprint arXiv:1805.06589*, 2018.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, 1996.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT 2003*, pages 68–86, 2003.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT 2018*, pages 395–427, 2018.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *International Conference on Cryptology and Information Security in Latin America*, pages 40–58. Springer, 2015.

[Cou06] Jean Marc Couveignes. Hard homogeneous spaces., 1997. *IACR Cryptology ePrint Archive*, 2006:291, 2006.

[Cox11] David A Cox. *Primes of the form $x^2 + ny^2$: Fermat, class field theory, and complex multiplication*, volume 34. John Wiley & Sons, 2011.

[CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In *CRYPTO '95*, pages 110–123, 1995.

[DNMQ12] Bernardo Machado David, Anderson CA Nascimento, and Jörn Müller-Quade. Universally composable oblivious transfer from lossy encryption and the mceliece assumptions. In *International Conference on Information Theoretic Security*, pages 80–99. Springer, 2012.

[dSGOPS18] Cyprien de Saint Guilhem, Emmanuela Orsini, Christophe Petit, and Nigel P Smart. Secure oblivious transfer from semi-commutative masking. *IACR Cryptology ePrint Archive*, 2018:648, 2018.

[DvdGMN08] Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. Oblivious transfer based on the mceliece assumptions. In *ICITS 2008*, pages 107–117, 2008.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[Gal12] Steven D Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[Hal05] Sean Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 468–474, 2005.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

[IG03] Ioannis Ioannidis and Ananth Grama. An efficient protocol for yao's millionaires' problem. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 6–pp. IEEE, 2003.

[JAC+17] David Jao, Reza Azarderakhsh, Matt Campagna, Craig Costello, Luca de Feo, Basil Hess, Amir Jalili, Brian Koziel, Brian Lamacchia, Patrick Longa, et al. Sike: supersingular isogeny key encapsulation, 2017. `https://sike.org/`.

[JDF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[Lin08] Andrew Y Lindell. Efficient fully-simulatable oblivious transfer. In *Cryptographers' Track at the RSA Conference*, pages 52–70. Springer, 2008.

[Lin17] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer, 2017.

[MCR19] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In *International Conference on Post-Quantum Cryptography*, pages 307–325. Springer, 2019.

[Mon87] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[NIS20] NIST. National institute of standards and technology, 2020. `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions`.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[OAYT19] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. A faster constant-time algorithm of CSIDH keeping two torsion points. *IACR Cryptology ePrint Archive*, 2019:353, 2019.

[Ode09] Goldreich Oded. Foundations of cryptography: Volume 2, basic applications. 2009.

[PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over gf (p) and its cryptographic significance (corresp.). *IEEE Transactions on information Theory*, 24(1):106–110, 1978.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571, 2008.

[Rab05] Michael O Rabin. How to exchange secrets with oblivious transfer., 1981. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptology ePrint Archive*, 2006:145, 2006.

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

[Vit19] Vanessa Vitse. Simple oblivious transfer protocols compatible with supersingular isogenies. In *AFRICACRYPT 2019*, pages 56–78, 2019.

[WZW03] Qian-Hong Wu, Jian-Hong Zhang, and Yu-Min Wang. Practical t-out-n oblivious transfer and its applications. In *International Conference on Information and Communications Security*, pages 226–237. Springer, 2003.

A Equivalence of The Square/Inverse/Reciprocal CSIDH Problem and The Computational CSIDH Problem

We will show the computational CSIDH problem is equivalent to the square variant with a quantum reduction. The order of the ideal class group can be computed with a quantum algorithm [Sho99, Hal05]. In Proposition 2.1 above, we have shown equivalence for the case that the order of the class group of the endomorphism ring is odd which is the case when $p = 3 \mod 4$. The remaining case is that the class number is even which happens when $p = 1 \mod 4$. In this case, the discriminant is $-4p$.

Note that the direct isogeny computation of a^e acting on E where a, E are given and $e = \Theta(\log(\#Cl))$ is avoided in the following proof, since these types of isogeny computations may not be polynomial-time [CLM+18, BFJ16], which would make the reduction non-polynomial-time. Let $(x)_y^{-1}$ denote the inverse of $x \mod y$.

Proposition A.1. *The square CSIDH problem is equivalent to the computational CSIDH problem if the order h of the group Cl is given.*

Proof. Let $p = 1 \mod 4$ so that the order h of the class group is even. Since the discriminant of the class group is $-4p$, by Proposition 3.11 of [Cox11], the 2-Sylow subgroup of the class group is of rank 1. Hence, the class group is isomorphic to $\mathbb{Z}_{2^t} \times \mathbb{Z}_{h'}$ for some $t, h' \in \mathbb{N}$ with h' being odd. Given the challenge $(E, a * E, b * E)$ and access to the oracle O, the goal is to find the curve $ab * E$.

Define the mapping $\chi : Cl \to Cl \times Cl$ where $\chi_{2^t}(\mathfrak{a}) = \mathfrak{a}^{h'(h')^{-1}_{2^t}}$, $\chi_{h'}(\mathfrak{a}) = \mathfrak{a}^{2^t(2^t)^{-1}_{h'}}$ and $\chi(\mathfrak{a}) = (\chi_{2^t}(\mathfrak{a}), \chi_{h'}(\mathfrak{a}))$. The image of χ is isomorphic to $\mathbb{Z}_{2^t} \times \mathbb{Z}_{h'}$. The mapping χ satisfies $\chi_{2^t}(x)\chi_{h'}(x) = x$. Given $\mathfrak{a} * E$, the pair $(\chi_{2^t}(\mathfrak{a}) * E, \chi_{h'}(\mathfrak{a}) * E)$ can be efficiently computed by Lemma 2.1 by using the oracle.

Run the following polynomial-time algorithm.

1. Compute $a'' * E$ and $b'' * E$ where $a'' = \chi_{h'}(a)$ and $b'' = \chi_{h'}(b)$ by Lemma 2.1.
2. From Proposition 2.1, given $a'' * E, b'' * E$, one can get $a''b'' * E$ since a'', b'' are of odd order.
3. Compute $(a''b'')^{\frac{h'+1}{2}} * E$ using Lemma 2.1. Denote $q = (a''b'')^{\frac{h'+1}{2}} \in Cl$. Note that $q^2 = a''b''$.
4. Generate a curve $g * E$ where $g \in Cl$ is of order 2^t. (See details below.)
5. Compute $a' * E$ and $b' * E$ where $a' = \chi_{2^t}(a)$ and $b' = \chi_{2^t}(b)$. Write $a' = g^{\sum a'_i 2^i}$ and $b' = g^{\sum b'_i 2^i}$ where $a'_i, b'_i \in \{0, 1\}$.
6. Obtain a'_0 by computing $a'^{2^{t'-1}} * E$ by Lemma 2.1. If it is E, then $a'_0 = 0$. Otherwise, $a'_0 = 1$.
7. Iteratively, for $j < t - 1$, assume $a'_0, ..., a'_{j-1}$ are known, then obtain a'_j by computing $a'^{2^{t-j-1}} * E$. If, with Lemma 2.1, the curve equals
$$g^{\sum_0^{j-1} 2^{i+t-j-1} a'_i} * E,$$
then $a'_j = 0$; otherwise, $a'_j = 1$.
8. Repeat Step 6 and Step 7 for b' to obtain $b'_i \in \{0, 1\}$.
9. Compute $g^{-\sum(a'_i+b'_i)2^i} * E$ using Lemma 2.1. (Note that $\chi_{2^t}((ab)^{-1}) * E = g^{-\sum(a'_i+b'_i)2^i} * E$.)
10. Compute $O(g^{-\sum(a'_i+b'_i)2^i} * E, q * E)$ to obtain $ab * E$.

In Step 4, the curve can be generated by sampling a random element $g_{pre} \in Cl$ and raising g_{pre} to the power of $2^{t-1}h'$. If it's the identity element in Cl, then restart. Otherwise, set $g * E$ to be $g_{pre}^{h'} * E$ by Lemma 2.1. If the sampling is random enough, then the success rate is 1/2 for each trial.

The relation between $a' * E$ and $g * E$ is computed in Step 6. Since g is of order 2^t, then $a'^{2^{t-1}} * E = g^{a'_0 2^{t-1}} * E$. Hence, a'_0 is 0 if and only if the outcome is E.

In Step 7, the idea of Step 6 is taken one step further to recover a'_j for $j = 1, ..., t - 1$ iteratively. This idea is known as the Pohlig-Hellman attack [PH78]. If $a'_0, ..., a'_{j-1}$ are known, raising $a' * E$ to the power of $t - j - 1$ eliminates $a_{j+1}, ..., a_{t-1}$ in the exponentiation of a' with the base g, since the order of g is 2^t. We can thereby find out a'_j through comparing. To be more specific, due to

$$a'^{2^{t-j-1}} * E = g^{\sum_0^j a'_i 2^{i+t-j-1}} * E,$$

we have $a'_j = 0$ if and only if

$$g^{\sum_0^{j-1} 2^{i+t-j-1} a'_i} * E = g^{\sum_0^{j} 2^{i+t-j-1} a'_i} * E.$$

The same reasoning holds for b. Hence, we can compute $g^{-\sum (a'_i + b'_i) 2^i} * E$, which is $\chi_{2^t}((ab)^{-1}) * E$.

In step 10, we invoke the oracle of the square CSIDH problem and get

$$\begin{aligned}
& O\left(g^{-\sum (a'_i + b'_i) 2^i} * E, q * E\right) \\
= & \left(g^{\sum (a'_i + b'_i) 2^i} q^2\right) * E \\
= & \left(\chi_{2^t}(ab)(a''b'')^{h'+1}\right) * E \\
= & \left(\chi_{2^t}(ab)(a''b'')\right) * E \\
= & \left(\chi_{2^t}(ab)\chi_{h'}(ab)\right) * E \\
= & ab * E.
\end{aligned}$$

□

With the reduction in the context (Proposition 2.2), we have shown equivalence between square, inverse, reciprocal variants. Therefore, in a generic CSIDH setting, we have the following relation

Computational CSIDH $=_{quantum}$ **Computational Inverse/Square/reciprocal CSIDH**.