



# Simple Two-Round OT in the Explicit Isogeny Model

Emmanuela Orsini<sup>1</sup> \*  and Riccardo Zanotto<sup>2</sup> \*\* 

<sup>1</sup> Bocconi University, Milano, Italy and imec-COSIC, KU Leuven, Leuven, Belgium

`emmanuela.orsini@kuleuven.be`

<sup>2</sup> CISP Helmholz Center for Information Security

`riccardo.zanotto@cispa.de`

**Abstract** In this work we apply the Type-Safe (TS) generic group model, recently introduced by Zhandry (2022), to the more general setting of group actions and extend it to the universal composability (UC) framework of Canetti (2000). We then relax this resulting model, that we call UC-TS, to define an *algebraic action* framework (UC-AA), where the adversaries can behave algebraically, similarly to the algebraic group model (AGM), but for group actions. Finally, we instantiate UC-AA with isogeny-based assumptions, obtaining the Explicit-Isogeny model, UC-EI, and show that, under certain assumptions, UC-EI is less restricting than UC-AGM.

We demonstrate the utility of our definitions by proving UC-EI security for the passive-secure protocol described by Lai et al. (2021), hence providing the first concretely efficient two-round isogeny-based OT protocol in the random oracle model against malicious adversaries.

## 1 Introduction

Oblivious transfer (OT), introduced by Rabin [Rab05], is a fundamental cryptographic primitive that plays a central role in modern cryptography. In particular, OT is sufficient and necessary for secure two-party and multiparty computation, and it is widely deployed in a number of efficient protocols [BLN<sup>+</sup>21, KOS16] and applications ranging from private set intersection [DCW13, PSZ14] to contract signing [EGL82]. The most commonly studied form of oblivious transfer is 1-out-of-2 OT: here a sender  $P_S$  holds two messages  $m_0, m_1$  and a receiver  $P_R$  holds a bit  $b$  corresponding to the sender's message  $m_b$  that it will receive as output of the protocol. The security requirement is that  $P_R$  should obtain  $m_b$  but no information about the other message  $m_{1-b}$  and  $P_S$  should learn nothing about the bit choice  $b$ .

Oblivious transfer can be built from a variety of assumptions: number-theoretic assumptions like Decisional Diffie-Hellman (DDH) [BM90, NP01, AIR01, PVW08, ZLWR13, CO15], and quadratic-residuosity (QR) [HK12]; and also from (presumed) post-quantum assumptions like coding-theory related assumptions [DvMN08, DDN14, DGH<sup>+</sup>20], lattice-based assumptions [PVW08, BD18, MS20], and isogeny-based assumptions [Vit19, BDGM19, dOPS20, LGd21].

*Security and round complexity.* Since oblivious transfer is often used as a building block inside bigger cryptographic schemes, its efficiency greatly influences the efficiency of these systems; in addition, even if there exists a variety of security models for OT, ideally we would like to achieve security in the *universal composability* (UC) framework described by Canetti [Can01], where security is maintained under concurrent general composition. Unfortunately, achieving efficient UC-secure OT protocols

---

\* Emmanuela Orsini was supported by CyberSecurity Research Flanders with reference number VR20192203. Her work was primarily carried out while she was affiliated with imec-COSIC.

\*\* Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

is not an easy task, especially if we want a protocol with a low round complexity. A straightforward way to achieve fully secure OT is by using zero-knowledge proofs to transform a passively secure protocol into an active one. However, this usually adds more rounds of communication to the passive protocol, therefore if the goal is to describe a protocol with a small number of rounds, e.g. 2 or 3, we need to adopt a different approach and focus on low constant-round protocols that are already actively secure.

A first barrier is given by the impossibility result of Goldreich and Oren [GO94] which states that two-round OT with simulation-based security cannot be achieved in the plain model, so we necessarily need to rely on setup assumptions such as a *common reference string* (CRS) or a *random oracle* (RO). In a nutshell, the main problem in a simulation-based proof is to extract the receiver’s input bit and then argue that the message  $m_{1-b}$  remains hidden.

Restricting our attention exclusively on (presumed) post-quantum assumptions, only few constructions for two-round maliciously secure OT are known, namely the general framework described by Peikert, Vaikuntanathan and Waters [PVW08] and the compiler described by Döttling et al. [DGH<sup>+</sup>20]. While theoretically interesting, both schemes are rather complex and inefficient. Therefore, we can pose the following natural question:

*Can we describe a concretely efficient UC-secure two-round oblivious transfer protocol from (presumed) post-quantum secure assumptions, and in particular from isogeny-based assumptions?*

*Why isogeny-based OT.* One of the main motivations of some isogeny-based cryptographic schemes is their resistance so far to cryptanalysis by quantum algorithms and their main advantage, compared to other post-quantum secure constructions, are relatively small keys and a rich mathematical structure.

The first isogeny-based cryptosystem was given by Couveignes in 1997 [Cou06] and independently re-discovered by Rostovtsev and Stolbunov [RS06]. They described a non-interactive key-exchange protocol where the public key is taken from the set of  $\mathbb{F}_q$ -isomorphism classes of ordinary elliptic curves over  $\mathbb{F}_q$ , whose endomorphism ring is a given order  $\mathcal{O}$  in an imaginary quadratic field. The main observation was that the commutativity of the ideal class group  $Cl(\mathcal{O})$ , which acts freely and transitively on this set, naturally leads to a key-exchange procedure à la Diffie-Hellman.

Later, to avoid an attack due to Childs et al. [CJS14] that exploits the commutativity of  $Cl(\mathcal{O})$ , Jao and De Feo [JD11] proposed the supersingular isogenies Diffie-Hellman (SIDH) key-exchange scheme, using supersingular elliptic curves, whose full-ring of endomorphism is non-commutative. Unfortunately, this scheme and related variants, like the NIST candidate SIKE [JAC<sup>+</sup>20], have been recently broken [CD22, MM22, Rob22] using Kani’s reducibility criterion [Kan97]. These attacks have no effect on another class of isogeny-based cryptographic schemes, namely CSIDH-based constructions. CSIDH stands for Commutative-SIDH and it was introduced by Castrick et al. [CLM<sup>+</sup>18], as an adaptation of the original Couveignes-Rostovtsev-Stolbunov protocol with supersingular curves.

Most of the known isogeny-based OT schemes can be directly derived by the above mentioned key-exchange protocols, exactly like the Chou and Orlandi [CO15] protocol can be seen as tweaks of two DDH-based key exchange schemes. These schemes are very simple, but, as happens for other OT constructions, they cannot easily be proven fully UC secure.

## 1.1 Our Contributions

In this work we consider the question above and provide a simple isogeny-based two-message OT protocol in a new computational model that we call *UC Explicit-Isogeny* (UC-EI). Like in UC-AGM described by Abdalla et al. [ABK<sup>+</sup>21], we cannot prove equivalence to standard UC, however we can show some evidence that our new framework is less restrictive than the corresponding UC-AGM. We conclude with a thoughtful comparison between several isogeny-based OT protocols.

We proceed to describe our contributions in more detail.

**Explicit isogeny.** In [CO15], Chou and Orlandi proposed a simple and efficient OT protocol, hereafter known as the CO protocol (we refer to Section A.1 for further details), based on DDH in the random oracle model. This scheme has been used as a blueprint for many subsequent constructions based on different hardness assumptions. However, proving its UC security is problematic: the main issue is that if a corrupt receiver never makes a query to the random oracle to decrypt its ciphertext, the simulator cannot extract the receiver’s choice bit and thus cannot query the functionality for the corresponding sender’s message and finish the simulation.

One solution to this problem is to require some sort of “proof of decryption”: some protocols, like [BDD<sup>+</sup>17], add one or two rounds of interaction to allow the simulator to extract and to complete the simulation of the corresponding functionality. This same approach was taken by Lai et al. [LGd21] to describe a concretely efficient 4-round isogeny-based UC-secure oblivious transfer in the CSIDH setting.

Another solution, proposed in [ABK<sup>+</sup>21], is given by using different models of computations, in which the extraction of the input bit is made almost automatic by the behaviour imposed on the adversary by the model. More concretely, Abdalla et al. introduced a new model of computation, UC-AGM, obtained by extending the *algebraic group model* (AGM) within the UC framework, and showed that some important protocols, like CO, can be proven secure in this model. We recall that AGM, introduced by Fuchsbauer, Kiltz and Loss [FKL18], takes the middle ground between the standard model and the *generic group model* (GGM), which has been proposed in different and not necessarily equivalent variants, like the ones described by Shoup [Sho97] and Maurer [Mau05]. Roughly, in AGM adversaries are allowed to see and use the structure of the group but are required to also give a representation of every output group element as a linear combination of the inputs.

*Drawback of the AGM.* While the AGM (and UC-AGM) enables easier proofs of security for protocols, the model also shows some limitations and gives rise to concerns. For example, in the AGM we are not able to sample group elements without access to an external oracle; it is limited to prime-order groups, and, in its original definition, it does not include pairing groups.

Another issue, highlighted concurrently by [KZZ22] and [Zha22], is that the AGM is incomparable to Shoup’s GGM, due to an informal requirement of the AGM. Katz et al. focus on the the questions of how to encode group elements and of what it means to “output a group element”; in particular, the authors prove that an algebraic adversary can be used to reduce a generic-hard problem to a generic-easy problem, thus showing a key weakness in the AGM’s definition. In the paper by Zhandry a solution to this issue is proposed by formally defining the AGM as a compiler for games in the *Type-Safe* (TS) generic-group model, which is a variant of Maurer’s GGM; the TS model restricts the type of games that can be described, and among those the AGM requirement is met.

There are also issues when trying to combine computational models and the UC framework; for example, [Zha22] showed that security in the TS model is equivalent to security in Shoup’s model

for *single-stage* games, which however cannot describe UC security. A more fundamental problem, specific to the UC-AGM [ABK<sup>+</sup>21], is that a composition theorem holds only if the algebraic adversary “does not mix” protocols, i.e., if it explains group elements of some sub-protocol using only group elements from the same sub-protocol as a base; this means that we can safely compose protocols only if they operate on independent groups.

*Our approach.* Informally, in this work, inspired by the effectiveness of UC-AGM for the CO protocol, we introduce a new computational model, which we call *Explicit Isogenies* (EI), where an adversary, instead of giving a representation for a group element, has to “explain” each supersingular elliptic curve  $E$  that it outputs by giving the hidden path used to produce such a curve.

Even if the issues mentioned above about AGM do not directly apply to our use case, in formalising our models we use the TS model introduced by Zhandry for the more general setting of *group actions*. Given this, we describe a new model that we call the *algebraic action model* (AAM), which can be considered as an equivalent of the TS model when compiled with AGM, but for group actions; we then extend it to the multiparty setting within the UC framework.

Informally, a group action  $\mathcal{G}$  consists of a group  $G$  and a set  $X$ , along with an *action*  $\star : G \times X \rightarrow X$ , such that for any  $g, g' \in G$  and  $x \in X$ , it holds that  $(gg') \star x = g \star (g' \star x)$ , and it allows to capture the exponentiation-only restriction of some isogeny settings. In [ADMP20], Alamati et al. introduced the notions of *effective-GA* (EGA) and *restricted-EGA* (REGA) to model cryptographic group actions and in particular CSIDH, where computing the group action efficiently is not possible for all group elements  $g \in G$ . To define our UC-TS model, we propose a new generalization of EGA and REGA to *hint-effective GA* (HEGA); our TS model for group actions models *generic actions*, but with a different approach from [MZ22], where “generic” algorithms operate independently of the representation of both group elements and set elements.

We then consider the restriction of UC-TS to algebraic adversaries, obtaining the UC-AA model. Loosely speaking, in our setting an adversary is said to be *algebraic* if, every time they output a set element  $y \in X$ , they must *explain* it as a group action  $y = x \star g$ , where  $x \in X$  was a set element already known; the adversary moreover has complete access to the group structure and representation. In our definitions, we only consider HEGA and we show that the CSIDH action group  $\mathcal{G}_{\text{tw}}$ , which also comprises twist, is indeed an example of HEGA (we refer to Section 2.4 for a brief overview of CSIDH). This finally defines our UC-EI.

Similarly to UC-AGM, also UC-AA (and consequently UC-EI) limits the capability of the adversary hence, like UC-AGM, it is less expressive than standard UC. Restricting to UC-EI, however, a well known open problem in isogeny-based cryptography is how to sample random supersingular elliptic curves without taking a random isogeny walk from an already known curve. The sampling problem has recently gained some attention [MMP22, BBD<sup>+</sup>22], but so far all the attempts to solve it have failed. Assuming the problem is hard implies that the CSIDH action group  $\mathcal{G}_{\text{tw}}$  is actually an unsampleable HEGA, and therefore restricting algorithms to be algebraic does not restrict the model. Notice this is different to what happens in AGM, where algebraic algorithms have to provide a representation of any group element they output, but it is not true that the only way for them to output a new group element is to derive it using group multiplication from known group elements.

On the other hand, like in UC-AGM, UC-EI composition theorem holds only if the adversary is “non-mixing”, i.e., an adversary attacking a protocol  $\phi$  can only send and use messages that can be explained using element wires from  $\phi$ . As noted in [ABK<sup>+</sup>21], this should not necessarily be seen as a limitation of UC-EI (and UC-AGM), but rather as a limitation of proofs in idealized models;

more discussion on this point can be found on the paper. Ways to overcome this issue, and consider also “mixing adversaries”, can be either proving multiple protocol executions simultaneously or proving security in extended settings like GUC (UC with global setup) [CDPW07].

**2-round isogeny-based OT.** Two-message OT is a very desirable primitive since it is complete for both 2-party and multiparty computation protocols. Once we established our UC-EI model, proving full security of the passively secure OT protocol given in [LGd21] is a relatively easy task. In this way, we obtain an efficient two-round protocol in the ROM with a trusted setup curve (TSC) based on CSIDH. In addition, we show how to eliminate the TSC requirement, but at the cost of adding an extra round of communication, so the resulting scheme is not round optimal. This latter protocol needs the same number of messages of the maliciously secure protocol described in [LGd21], but with less computation and without TSC. We refer to Section 5.3 for a detailed comparison between different isogeny-based protocols.

## 1.2 Other Related Work

Another important line of work aims to construct two-message oblivious transfer with a slightly weaker form of security, namely statistically sender-private OT (SSP OT) [NP01, AIR01]. In this setting, different schemes based on different quantum secure assumptions are known, like [BD18, DGI+19, MS20, ADMP20].

**Concurrent work.** The paper [BMM+22] introduces some new OT protocols based on isogenies; in particular, the authors show how to build UC-secure round-optimal protocols based on the computational CSIDH assumption, both in the plain model and in the setup model. This is an important result, since round-optimal protocols were known only from the decisional CSIDH problem [ADMP20]. More details about their constructions and efficiency can be found in Section 5.3.

In [DHK+23], the authors introduce generic models for group actions and prove generic hardness results and equivalence between different assumptions in a quantum setting; in particular, their Algebraic Group Action Model is almost identical to the one proposed by us, and they both differ from the one first proposed by [MZ22] in the sense that we both only encode set elements and otherwise give full access to the group to the adversary. However, we follow [Zha22] to first describe a type-safe generic model which formally delimits the games that can be described in our Algebraic Action Model, while [DHK+23] poses the same informal constraint as in the AGM that “input that is not a set element  $x \in X$  does not depend on set elements”. Another difference is that [DHK+23] only considers abelian group actions (and in particular of the concrete group  $\mathbb{Z}/n\mathbb{Z}$ ), and the authors have to model externally the twisting property of CSIDH, while we embed it in a bigger group action, which is however not abelian nor regular.

## 2 Preliminaries

For a set  $S$ , we denote by  $a \leftarrow \$ S$  the process of drawing  $a$  from  $S$  with a uniform distribution on  $S$ . If  $\mathcal{D}$  is a probability distribution, we denote by  $a \leftarrow \$ \mathcal{D}$  the process of drawing  $a$  with the given probability distribution. For a probabilistic algorithm  $\mathcal{A}$ , we denote by  $a \leftarrow \$ \mathcal{A}$  the process of assigning  $a$  the output of algorithm  $\mathcal{A}$ , with the underlying probability distribution being determined by the random coins of  $\mathcal{A}$ .

We write  $\approx$  (resp.  $\approx_s$ ) to denote computational (resp. statistical) indistinguishability between probabilistic distributions.

## 2.1 Cryptographic Group Actions

Group actions have recently been getting a lot of attention for their use in cryptography, starting from the “Hard Homogeneous Spaces” by Couveignes [Cou06], which is a precursor of CSIDH. Here we will follow the formalization by Alami et al. [ADMP20].

**Definition 1 (Group action).** *A group  $G$  is said to act on a set  $X$  if there is a map  $\star : G \times X \rightarrow X$  that satisfies the following two properties:*

IDENTITY. *If  $e$  is the identity of  $G$ , then,  $\forall x \in X$ , we have  $e \star x = x$*

COMPATIBILITY. *For any  $g, h \in G$  and any  $x \in X$ , then  $(gh) \star x = g \star (h \star x)$ .*

A group action as described in the previous definition is usually denoted by  $\mathcal{G} = (G, X, \star)$ . The standard notion of cryptographic group actions is given by *effective* group action (EGA). Roughly, an EGA  $(G, X, \star)$  is such that all the well-defined group operations and group action operations are efficiently computable, sampling random group elements is efficient and set elements are uniquely represented. It is also possible to endow group actions with different hardness properties like one-way EGA, weak-unpredictable EGA, weak pseudorandom EGA [ADMP20].

EGA is usually too powerful to capture isogeny-based assumptions, therefore, to model isogeny-based protocols, [ADMP20] provides the definition of *restricted effective group action* (REGA), where it is possible to only evaluate the action of a generating set of small cardinality.

## 2.2 Elliptic Curves, Isogenies, Endomorphisms

An elliptic curve is a smooth curve of genus one with a distinguished rational point. More concretely, an elliptic curve  $E$  defined over a field  $K$  (denoted  $E/K$ ) is the set of points satisfying the Weierstrass equation  $y^2 = x^3 + ax + b$ , with  $a, b \in K$ , with an additional “point at infinity”. The points of an elliptic curve, denoted by  $E(K)$ , have an abelian and algebraic group structure given by intersecting lines with the curve. Given two elliptic curves  $E_1$  and  $E_2$  over  $K$ , an *isogeny*  $\psi$  between  $E_1$  and  $E_2$  is a non-constant morphism between them. The degree of an isogeny is its degree as a rational map; an isogeny is said to be *separable* if its degree is equal to the size of its kernel. Given a finite subgroup  $G$  of  $E$ , there exists a separable isogeny  $\psi : E \rightarrow E/G$ , with  $\ker \psi = G$ , which is unique up to isomorphism. Both the isogeny  $\psi$  and image  $E/G$  can be computed from the kernel using Vélu’s formulas [Vél71], whose efficiency depends on the smoothness of the isogeny degree.

An *endomorphism* of an elliptic curve  $E$  is an isogeny from  $E$  to itself. The set  $\text{End}(E)$  of endomorphisms of  $E$ , together with the zero map, is a ring. When  $E$  is defined over a finite field, the endomorphism ring of  $E$  is either an order in a quadratic field, in which case we say  $E$  is *ordinary*, or a maximal order in a quaternion algebra and  $E$  is called *supersingular*. For more background on elliptic curves, isogenies and their use in cryptography we refer to standard resources [Sil09, Feo17].

## 2.3 SIDH

The protocol proposed in [JD11] is the first key-exchange based on isogenies of supersingular elliptic curves, and it gave birth to the field of isogeny-based cryptography. Unfortunately, the SIDH protocol and its underlying assumption have recently been completely broken by a series of efficient attacks [CD22, MM22, Rob22], which are based on higher dimension abelian varieties.

<b>Protocol SIDH</b>	
<p><b>Alice</b></p> $a \leftarrow_{\$} \mathbb{Z}/\ell_A^{e_A} \mathbb{Z}$ $A = P_A + [a]Q_A$ $\phi_A : E_0 \rightarrow E_A = E_0/\langle A \rangle$	<p><b>Bob</b></p> $b \leftarrow_{\$} \mathbb{Z}/\ell_B^{e_B} \mathbb{Z}$ $B = P_B + [b]Q_B$ $\phi_B : E_0 \rightarrow E_B = E_0/\langle B \rangle$
$\xrightarrow{E_A, (\phi_A(P_B), \phi_A(Q_B))}$ $\xleftarrow{E_B, (\phi_B(P_A), \phi_B(Q_A))}$	
$B' = \phi_B(P_A) + [a]\phi_B(Q_A)$ $\phi'_A : E_B \rightarrow E_{BA} = E_B/\langle A' \rangle$ $s = j(E_{BA})$	$A' = \phi_A(P_B) + [b]\phi_A(Q_B)$ $\phi'_B : E_A \rightarrow E_{AB} = E_A/\langle B' \rangle$ $s = j(E_{AB})$

**Figure 1.** The SIDH protocol

The high level idea of the SIDH protocol is intuitively given by the following diagram:

$$\begin{array}{ccc}
 E & \xrightarrow{\phi} & E/\langle R \rangle \\
 \downarrow \psi & & \downarrow \\
 E/\langle S \rangle & \longrightarrow & E/\langle R, S \rangle
 \end{array}$$

where  $E$  is a supersingular elliptic curve, the points  $R$  and  $S$  are the secrets of Alice and Bob, the two quotients  $E/\langle R \rangle$  and  $E/\langle S \rangle$  are the exchanged values and finally  $E/\langle R, S \rangle$  is the shared secret. The two isogenies  $\phi$  and  $\psi$  are computed by a random walk in different-degree isogeny graphs. A more formal description of the SIDH key-exchange protocol is given in Figure 1.

The security of the protocol stands in the fact that it is hard for an attacker to recover the secret isogeny  $\phi$ , since the following problem is considered hard:

**Problem 1 (IsogenyPath)** *Let  $E, E'$  be two isogenous curves. Find an isogeny  $\phi : E \rightarrow E'$ .*

However, the protocol needs more information to work, in particular the action of  $\phi$  and  $\psi$  on specific torsion subgroups of  $E$ , as it can be seen in Figure 1. Indeed, the actual hardness assumption for SIDH is the following.

**Problem 2 (Computational Supersingular Isogeny Problem (CSSI))** *Let  $E_0/\mathbb{F}_{p^2}$  be a supersingular elliptic curve, and  $\{P_A, Q_A\}, \{P_B, Q_B\}$  be fixed basis for  $E_0[\ell_A^{e_A}]$  and  $E_0[\ell_B^{e_B}]$  respectively; let  $\phi_A : E_0 \rightarrow E_A$  an isogeny with kernel  $K_A = \langle P_A + aQ_A \rangle$ . Given  $E_A$  and  $\phi_A(P_B), \phi_A(Q_B)$ , find a generator of  $K_A$ .*

The key weakness of the protocol is thus the additional torsion point information  $\phi_A(P_B), \phi_A(Q_B)$ , which was already exploited in [Pet17, dQKL<sup>+</sup>21] against very peculiar parameter sets, and by the recent attacks [CD22, MM22, Rob22] against all possible instantiations of this protocol.

## 2.4 CSIDH

In [CLM<sup>+</sup>18], the authors propose the first post-quantum abelian group action, from which they derive a key-exchange primitive, called CSIDH.

They consider a supersingular elliptic curve  $E$  over  $\mathbb{F}_p$ , so that its  $\mathbb{F}_p$ -rational endomorphism ring  $\text{End}_p(E)$  is an order  $\mathcal{O}$  in a quadratic imaginary field. If  $\mathfrak{a}$  is a non-zero ideal in  $\mathcal{O}$ , then it defines a finite subgroup  $E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker(\alpha)$ , where each  $\alpha$  is identified with its image in  $\text{End}(E)$ . We can consider the quotient isogeny  $\psi : E \rightarrow E' = E/E[\mathfrak{a}]$  with kernel  $\mathfrak{a}$ . This isogeny, as well as its codomain, is well-defined up to isomorphism. If  $\mathfrak{a} = (\alpha)$  is a principal ideal, then  $\psi \cong \alpha$  and  $E/E[\mathfrak{a}] \cong E$ . Denoting by  $\mathcal{E} = \text{Ell}_p(\mathcal{O})$  the set of curves having  $\mathcal{O}$  as their  $\mathbb{F}_p$ -endomorphism ring, we have a free and transitive action of  $Cl(\mathcal{O}) \times \mathcal{E} \rightarrow \mathcal{E}$  given by  $\mathfrak{a} \star E := E/E[\mathfrak{a}]$ .

The main idea of CSIDH is to pick a prime of the form  $p = 4\ell_1 \dots \ell_n - 1$ , with  $\ell_i$  small odd primes. In addition, they fix  $E_0 : y^2 = x^3 + x$ , which is supersingular when  $p \equiv 3 \pmod{4}$ . This curve has  $\text{End}_p(E_0) = \mathbb{Z}[\pi] \cong \mathbb{Z}[\sqrt{-p}]$ , where  $\pi$  is the  $\mathbb{F}_p$ -Frobenius map of  $E$ .

Since the characteristic polynomial of the Frobenius map is  $\pi^2 + p = 0$ , when reduced modulo  $\ell_i$  it becomes  $\pi^2 - 1 \equiv 0 \pmod{\ell_i}$ , given that  $p \equiv -1 \pmod{\ell_i}$ . In particular, this means that  $\ell$  splits as the product of  $\mathfrak{l}_i = (\ell_i, \pi - 1)$  and  $\bar{\mathfrak{l}}_i = (\ell_i, \pi + 1)$  inside  $\mathcal{O}$  (primes that are split are called *Elkies* primes).

The very peculiar choice of the prime  $p$  implies that the evaluation of the action  $\mathfrak{l}_i \star E$  is very easy: the kernel of the corresponding isogeny  $E[\mathfrak{l}_i]$  is the intersection of  $\ker[\ell_i]$  and  $\ker(\pi - 1)$ , which is the  $\mathbb{F}_p$ -rational  $\ell_i$ -torsion subgroup. By computing it and using Vélu's formulas, we can compute the isogeny  $\varphi_{\mathfrak{l}_i}$ .

For computing  $\mathfrak{l}_i^{-1} \star E$  we can either compute the  $\mathbb{F}_{p^2}$ -rational  $\ell_i$ -torsion or we can use the fact that  $(\mathfrak{a} \star E)^t \cong \mathfrak{a}^{-1} \star E^t$ , where  $E^t$  is the quadratic twist of  $E$ . For more details on evaluating the class group action, we refer to [CLM<sup>+</sup>18, section 8].

Another key aspect of CSIDH is that it uses curves in Montgomery form. Indeed, we have the following proposition.

**Proposition 1.** *Let  $p \geq 5$  be a prime with  $p \equiv 3 \pmod{8}$ , and let  $E/\mathbb{F}_p$  be a supersingular elliptic curve. Then  $\text{End}_p(E) = \mathbb{Z}[\pi]$  if and only if there exists  $A \in \mathbb{F}_p$  such that  $E$  is  $\mathbb{F}_p$ -isomorphic to the curve  $E_A : y^2 = x^3 + Ax^2 + x$ . Moreover, such  $A$  is unique.*

This means that it is sufficient to use the  $A$  coefficient as the public key, instead of a  $j$ -invariant and then having to check that it has the correct endomorphism ring. The only check needed is that  $A \notin \{\pm 2\}$  (otherwise  $E_A$  is not even smooth), and that  $E_A$  is supersingular. Furthermore, it is very easy to see that in this setting the twist of  $E_A$  is simply  $E_A^t \cong E_{-A}$ .

*CSIDH assumptions.* We list below the main hardness assumptions we use in our protocols; these are mainly defined and studied in [Fel19] and [LGd21].

**Problem 3 (Vectorization, or DLog)** *Given curves  $(E, r \star E)$  with  $E \in \mathcal{E}, r \in Cl$ , the problem asks to find said element  $r$ .*

Notice that since the action is regular, the generic vectorization problem is equivalent to the one with  $E = E_0$ .

**Problem 4 (Computational CSIDH, or CDH)** *Given curves  $(E, r \star E, s \star E)$  in  $\mathcal{E}$  with  $r, s \in Cl$ , the problem asks to find  $E' \in \mathcal{E}$  such that  $E' = rs \star E$ .*



**Problem 5 (Computational Inverse CSIDH)** Given curves  $E, r \star E$  in  $\mathcal{E}$  with  $s \in Cl$ , the problem asks to find  $E' \in \mathcal{E}$  such that  $E' = r^{-1} \star E$ .

Given that  $E_0^t = E_0$ , we have that  $(\mathfrak{a} \star E_0)^t = \mathfrak{a}^{-1} \star E_0$ , so the above problem is easy in the special case of  $E = E_0$ .

**Problem 6 (Computational Reciprocal CSIDH)** Given  $E \in \mathcal{E}$ , first the adversary chooses and commits to  $X \in \mathcal{E}$ , then it receives the challenge  $s \star E, s \in Cl$ . The adversary wins if it can compute  $(s \star X, s^{-1} \star X)$  w.r.t.  $X$ .

We have the following reductions between these problems.

**Proposition 2.** *The computational reciprocal CSIDH problem is equivalent to the computational inverse CSIDH problem.*

**Proposition 3** ([Fel19], [LGd21]). *If the group  $Cl$  has known order, the computational CSIDH problem is equivalent to the computational inverse problem.*

This also means that the two problems are *quantumly* equivalent, since the computation of the class group can be done in quantum polynomial time.

Another fundamental quantum equivalence is the following.

**Theorem 1** ([GPSV18], [Wes22a], [MZ22]). *The vectorization problem and the computational CSIDH problem are quantumly equivalent.*

*Sampling ideals in the class group.* The group structure of  $Cl(\mathcal{O})$  is not generally known, and more importantly we don't know how to efficiently evaluate the action of generic ideals. This is why CSIDH is a REGA [ADMP20]. In particular, we know how to evaluate the action of the ideals  $\mathfrak{l}_i$  corresponding to the Elkies primes  $\ell_i$  of the factorization of  $p + 1$ , and so of all ideals of the form  $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$  for small exponents  $e_i$ . This is the reason of how the peculiar form of  $p$  was chosen, and the authors of [CLM<sup>+</sup>18] give the following heuristic lemma.

**Lemma 1 (Informal).** *Let  $\mathcal{D}_m$  be the distribution on  $Cl(\mathcal{O})$  given by sampling  $(e_1, \dots, e_n)$  uniformly at random from  $[-m, m]^n$  and outputting  $\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{e_i}$ . Then, if  $(2m + 1)^n \geq |Cl(\mathcal{O})| \approx \sqrt{p}$ , the distribution  $\mathcal{D}_m$  is statistically close to the uniform distribution on  $Cl(\mathcal{O})$ .*

We will use this result, and the following useful lemma that holds for the case of  $p = 3 \pmod{4}$  and standard CSIDH settings.

**Lemma 2** ([LGd21]). *Given a curve  $E \in \mathcal{E}$  and a distribution  $\mathcal{D}$  on  $Cl$ , let  $\mathcal{D} \star \mathcal{E}$  be the distribution of  $a \star E$  for  $a \leftarrow \mathcal{D}$ , and let  $(\mathcal{D} \star E)^t$  be the distribution on  $\mathcal{E}$  of  $(a \star E)^t$  for  $a \leftarrow \mathcal{D}$ . If  $\mathcal{D}$  is statistically indistinguishable from the uniform distribution on  $Cl$ , then  $\mathcal{D} \star E$  and  $(\mathcal{D} \star E)^t$  are statistically indistinguishable from the uniform distribution on  $\mathcal{E}$ .*

## 2.5 Other Computational Assumptions

The main problem of isogeny based cryptography is the *Endomorphism Ring Problem* (**EndRing**), which can be stated as follows.

**Problem 7 (EndRing)** *Given a prime  $p$  and a supersingular elliptic curve  $E$  over  $\mathbb{F}_{p^2}$ , compute  $\text{End}(E)$ .*

Since  $\text{End}(E)$  is a rank 4 lattice for supersingular curves, finding even one extra endomorphism that is not “trivial” (i.e. a combination of scalar multiplications and the Frobenius) seems to be very hard.

One of the most important results for isogeny based cryptography is the following theorem that relates `EndRing` to `IsogenyPath` (Problem 1 in Section 2.3).

**Theorem 2** ([Wes22b]). *The problems `EndRing` and `IsogenyPath` are equivalent, under the generalised Riemann hypothesis (GRH).*

We can also try to understand the relationship between CSIDH, where curves and isogenies are constrained to be  $\mathbb{F}_p$ -rational, and the general `EndRing` problem. In [CPV20], the authors describe an efficient algorithm that, given two curves  $E_1, E_2$  over  $\mathbb{F}_p$  and their full endomorphism rings, outputs a class group element  $\mathfrak{a}$  such that  $\mathfrak{a} \cdot E_1 = E_2$ . The problem is that this ideal usually doesn’t have a smooth norm, so its action cannot be efficiently computed. In order to smoothen the norm, we need to find relations in  $Cl(\mathcal{O})$  and then run lattice reduction algorithms, which greatly increase complexity. This previous result can be made effective, and we actually have the following equivalence.

**Theorem 3** ([Wes22a]). *`EndRing` and the effective CSIDH vectorization problem are equivalent, under GRH.*

## 2.6 Overview of the UC Framework

We present here a semi-formal overview of the universally composable (UC) model of security established by Canetti [Can01]. The UC framework allows for defining security properties tasks so that security is maintained under general composition with unbounded number of instances of arbitrary protocols running concurrently.

Protocols that aim to achieve security in this model are defined in three steps. First, the protocol and its execution in the presence of an adversary are formalized, this represents the real-life model which we also call the *real world*. Next, an ideal process for executing the task is defined; its role is to act as a trusted party by separately receiving the input of each party, honestly computing the result of the protocol internally and returning the output assigned to each party. In this ideal world, the parties do not communicate with one another but instead solely rely on the ideal functionality to provide them with their output. Finally, we say that the protocol in question UC-realizes the ideal functionality if running the protocol is equivalent to emulating the ideal functionality. Below, we provide a brief discussion with additional formal details.

*Real model.* An execution of a protocol  $\pi$  in the real model consists of  $n$  PPT *interactive Turing machines* (ITMs)  $P_1, \dots, P_n$  representing the computing parties. Each party is uniquely determined by a *party identifier* (ID) that is used to distinguish between participants of the same protocol instance. We also have two additional ITMs, an adversary  $\mathcal{A}$ , describing the behaviour of the corrupted parties and an environment  $\mathcal{Z}$ , representing the external network environment in which the protocol operates. The environment gives inputs to the honest parties, receives their outputs, and can communicate with the adversary at any point during the execution. The adversary controls the operations of the corrupted parties and the delivery of messages between the parties. In more details, when the environment is activated, it can read the output tapes of all honest parties and  $\mathcal{A}$ , and it can activate a subset of the parties and  $\mathcal{A}$  by writing `input` messages on their input tapes. Parties, once activated, can perform local computation, write on their output tape, send

messages to other parties and send **subroutine output** messages to  $\mathcal{Z}$ . The adversary  $\mathcal{A}$  can send **backdoor** messages to  $\mathcal{Z}$  and all parties, and receive **backdoor** messages from all parties. We say that the adversary is *passive* (or *semi-honest*) if it always instructs the corrupt parties to follow the protocol; while we say that the adversary is malicious if it may instruct the corrupt parties to arbitrarily deviate from the protocol's instructions. We let  $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z, \mathbf{r})$  denote  $\mathcal{Z}$ 's output on input  $z$  and security parameter  $\kappa$ , after interacting with  $\mathcal{A}, P_1, \dots, P_n$  running the protocol  $\Pi$  with random tape  $\mathbf{r}$ .

*Ideal model.* In the ideal protocol  $\text{IDEAL}_{\mathcal{F}}$ , we have  $n$  dummy parties  $P_1, \dots, P_n$  which interact with an ideal functionality  $\mathcal{F}$  in a simple way: they pass their private inputs to  $\mathcal{F}$  and wait for it to return their assigned output. There is also an ideal-adversary  $\mathcal{S}$  which is responsible for the delivery of messages. The ideal functionality  $\mathcal{F}$  defines the desired behaviour of the computation, playing the role of a trusted third party, and can communicate with the ideal adversary  $\mathcal{S}$  by providing and receiving **backdoor** information. Finally, the same environment  $\mathcal{Z}$  is present in the ideal world.  $\mathcal{Z}$  also prescribes the inputs and observes the outputs of all parties. We only consider *static* corruptions, hence the set of corrupt parties is fixed before the start of the computation and is known to  $\mathcal{F}, \mathcal{Z}$  and  $\mathcal{S}$ . We let  $\text{EXEC}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}(\kappa, z, \mathbf{r})$  denote  $\mathcal{Z}$ 's output on input  $z$  and security parameter  $\kappa$ , after interacting with  $\mathcal{S}$  and dummy parties  $P_1, \dots, P_n$  that interact with  $\mathcal{F}$  using random tape  $\mathbf{r}$ .

*UC emulation.* One of the main concept of the UC framework is that of UC-emulation. Informally, it involves two protocols, say  $\pi$  and  $\phi$ , and we say that  $\pi$  *UC-emulates*  $\phi$  ( $\pi \sim \phi$ ), if for every efficient adversary  $\mathcal{A}$  in an execution of  $\pi$ , there is an efficient ideal adversary  $\mathcal{S}$  in an execution of  $\phi$ , such that no efficient environment  $\mathcal{Z}$  can distinguish an execution of  $\pi$  with  $\mathcal{A}$  and an execution of  $\phi$  with  $\mathcal{S}$ , i.e.  $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{EXEC}_{\phi, \mathcal{S}, \mathcal{Z}}$ . In particular, it implies that  $\pi$  can be safely used on behalf of  $\phi$  without compromising security.

Since the security of protocols is defined by comparing the “real” protocol execution with an “ideal” one, we can instantiate the concept of protocol emulation with the very special case of  $\phi$  being an ideal protocol  $\text{IDEAL}_{\mathcal{F}}$  for the functionality  $\mathcal{F}$ . Therefore, we say that  $\pi$  UC-emulates  $\mathcal{F}$  if we can infer that  $\pi$  does not leak any other information to an adversary than  $\mathcal{F}$  would have, and hence *securely realizes* the given task no matter how many other instances of  $\pi$  and/or other protocols are executed concurrently. In this case we write  $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{EXEC}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ .

**Definition 2.** We say that a protocol  $\pi$  UC-realizes an ideal functionality  $\mathcal{F}$  if  $\pi$  UC-emulates the ideal protocol  $\text{IDEAL}_{\mathcal{F}}$ .

*Hybrid model.* When a protocol  $\pi$  uses an ideal functionality  $\mathcal{G}$  as a sub-routine, the UC-framework considers the  $\mathcal{G}$ -hybrid model. In this case the parties, in both real and ideal world, have access to a copy of the ideal functionality  $\mathcal{G}$ . In the real world, this is an independent trusted party that executes the functionality honestly. In the ideal world,  $\mathcal{S}$  executes an internal copy of the functionality  $\mathcal{G}$  and only interacts with  $\mathcal{F}$ . An important property of the UC framework is that the ideal functionality  $\mathcal{G}$  in a  $\mathcal{G}$ -hybrid model can be replaced with a protocol  $\rho$  that UC-realizes  $\mathcal{G}$ . More concretely, let  $\rho$  be a protocol that securely realizes  $\mathcal{G}$  and let  $\pi^\rho$  be identical to  $\pi$  with the exception that the interaction with each copy of  $\mathcal{G}$  is replaced with an interaction of a separate instance of  $\rho$ . Then  $\pi$  and  $\pi^\rho$  have essentially the same input/output behaviour. In particular, if  $\pi$  securely realizes  $\mathcal{F}$  in the  $\mathcal{G}$ -hybrid model, then  $\pi^\rho$  securely realizes  $\mathcal{F}$  in the standard model, i.e., without access to  $\mathcal{G}$  and any other functionalities. In the following, we informally state the composition theorem.

### Functionality $\mathcal{F}_{\text{OT}}$

The functionality runs with a receiver  $P_R$ , a sender  $P_S$  and an adversary  $\mathcal{S}$

1. On input (**receive**,  $sid, \sigma$ ) from  $P_R$ , if no message with the same  $sid$  has been stored, store (**receive**,  $sid, \sigma$ ).
2. On input (**send**,  $sid, (m_0, m_1)$ ) from  $P_S$ , if no message with the same  $sid$  has been stored, store (**send**,  $sid, (m_0, m_1)$ ).
3. If  $\mathcal{S}$  sends **abort**, forward **abort** to the honest parties. Otherwise, on input (**deliver**,  $sid$ ) from the adversary, if there have been stored both messages (**receive**,  $sid, \sigma$ ) and (**send**,  $sid, (m_0, m_1)$ ), send (**output**,  $sid, m_\sigma$ ) to  $P_R$ .

**Figure 2.** Oblivious transfer functionality

### Functionality $\mathcal{F}_{\text{RO}}$

The functionality runs with a receiver  $P_R$ , a sender  $P_S$  and an adversary  $\mathcal{S}$ . It is parametrized by a domain  $\mathcal{D}$  and range  $\mathcal{R}$ . It keeps a list  $L$  of pairs of values, which is initially empty and proceeds as follows:

1. Upon receiving a query  $m \in \mathcal{D}$ , if there is a pair  $(m, k') \in L$ , set  $k = k'$ ; otherwise choose  $k \leftarrow \mathcal{R}$  and store  $(m, k)$  in  $L$ .
2. Output  $k$ .

**Figure 3.** Random oracle functionality

### Functionality $\mathcal{F}_{\text{TSC}}$

The functionality runs with a receiver  $P_R$ , a sender  $P_S$  and an adversary  $\mathcal{S}$

- Upon activation, sample  $t \leftarrow \mathcal{C}l$  and output the curve  $t \star E_0$ .

**Figure 4.** Trusted setup functionality

**Theorem 4** ([Can01]). *Let  $\pi$  be a protocol that UC-securely realizes the ideal functionality  $\mathcal{F}$  in the  $\mathcal{G}$ -hybrid model, let  $\phi$  be a protocol that UC-securely realizes the ideal functionality  $\mathcal{G}$ , then the protocol  $\pi^\phi$ , obtained by replacing each call to the ideal functionality  $\mathcal{G}$  in  $\pi$  with a call to the sub-protocol  $\phi$ , securely realizes  $\mathcal{F}$  in the standard model.*

## 2.7 Functionalities

In this work we will make use of relatively standard functionalities. The main OT functionality,  $\mathcal{F}_{\text{OT}}$ , described in Figure 2, is a standard 2-party functionality.

We will also need a random oracle functionality,  $\mathcal{F}_{\text{RO}}$ , as described in Figure 3. It initially contains an empty list  $L$ , then, each time it receives a query  $m$  from a fixed domain  $\mathcal{D}$ , it checks if the queried value  $m$  is already present in the list  $l$ . If this is the case, the functionality output the pair  $(m, k)$  in  $L$ ; otherwise, it samples a random  $k \leftarrow \mathcal{R}$ , outputs  $(m, k)$ , and stores that pair in  $L$ .

Finally, we will make use of a trusted setup  $\mathcal{F}_{\text{TSC}}$ , described in Figure 4, that fixes a starting curve  $E \in \mathcal{E}$ . Note that it outputs  $E = t \star E_0$ , but not  $t$  which maps  $E_0$  to  $E$ .

## 3 A Type-Safe Model for Group Actions

In this section, we first propose a generalization of both EGAs and REGAs, called *hint-effective group action* (HEGA), and secondly introduce a variant of Zhandry’s type-safe (TS) model for the setting of group actions. We then describe the *algebraic action model* (AAM) as a compiler,

similarly to the AGM but for group actions, where any adversary must explain any set element that it outputs with the group element that has generated it. Finally, we study this model in the UC framework.

### 3.1 Hint-Effective Group Actions

**Definition 3 (HEGA).** A group action  $(G, X, \star)$  is hint-effective if the following properties are satisfied:

1. The group  $G$  is finite, and there is an efficiently sampleable distribution  $\mathcal{D}_G$  on  $G$ . Moreover, sampling from this distribution produces also a hint  $e$ , and we write that as  $(g, e) \leftarrow_{\S} \mathcal{D}_G$ .
2. The set  $X$  is finite, and there are efficient algorithms for membership testing and computing a unique representation.
3. There is a distinguished and known element  $x_0 \in X$ , called the origin.
4. There exists an efficient algorithm such that for any  $(g, e) \leftarrow_{\S} \mathcal{D}_G$  and any  $x \in X$  computes  $g \star x$ , eventually using the hint  $e$ .

We collect these parameters in  $\mathcal{G} = (G, X, \star, \mathcal{D}_G, x_0)$ .

Notice that an EGA is an HEGA where the hints are empty and group operations on  $G$  are also efficient, while a REGA is an HEGA where the hint is the exponent vector with respect to the chosen generating set. As the author notice themselves, all protocols of [ADMP20] built from EGAs can also be instantiated from HEGAs, as soon as  $\mathcal{D}_G$  is statistically close to the uniform distribution on  $G$ .

**Computational assumptions on HEGAs.** We can define computational assumptions on HEGAs exactly as for EGAs in [ADMP20].

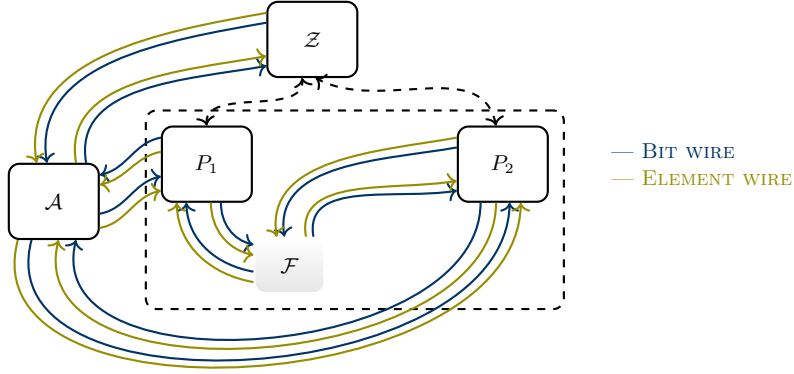
**Definition 4 (One-Way).** An HEGA  $\mathcal{G} = (G, X, \star, \mathcal{D}_G, x_0)$  is one-way if the family of functions  $\{f_x : G \rightarrow X\}$  is  $(\mathcal{D}_G \star x_0, \mathcal{D}_G)$ -one-way, where  $f_x(g) = g \star x$ .

Notice that  $f_x$  are efficiently computable, since the  $g$ s are sampled with hints from  $\mathcal{D}_G$ . More explicitly, we can define the “discrete logarithm” analogue for group actions as follows. Note that we do not restrict ourselves to regular actions, so there might be more than one possible answer.

**Problem 8 (DLog-HEGA)** Given  $x = g \star x_0$  for  $(g, e) \leftarrow_{\S} \mathcal{D}_G$ , compute any  $g' \in G$  such that  $x = g' \star x_0$ .

Similarly, it is possible to define pseudorandomness property, but we do not actually need it for our purpose. We finally describe an important property of a group action, namely the inability to sample directly from  $X$  without using group elements.

**Definition 5 (Unsampleable HEGA).** An HEGA  $\mathcal{G} = (G, X, \star, \mathcal{D}_G, x_0)$  is said to be unsampleable if for any PPT algorithm  $\mathcal{A}_G(x_1, \dots, x_n)$  that outputs a set element  $x \in X$ , there exists a PPT algorithm  $\mathcal{A}'_G$  that outputs  $(g, e)$  such that  $x = g \star x_i$  for some  $i \in \{0, 1, \dots, n\}$ .



**Figure 5.** Type-safe protocol  $\pi$  in the  $\mathcal{F}$ -hybrid model. The environment  $\mathcal{Z}$  provides the inputs and reads the outputs of the main parties;  $\mathcal{Z}$  and  $\mathcal{A}$  interacts freely. Note neither  $\mathcal{Z}$  nor  $\mathcal{A}$  have access to the functionality  $\mathcal{F}$ .

### 3.2 The Type-Safe Model

Following Zhandry’s type-safe model [Zha22], we define a similar model in the context of group actions, and in particular HEGAs.

**Definition 6.** Let  $\mathcal{G} = (G, X, \star, \mathcal{D}_G, x_0)$  be an HEGA. An algorithm  $\mathcal{A}$ , given as a circuit, is said to be type-safe w.r.t.  $\mathcal{G}$  (written as  $TS_{\mathcal{G}}$ ) if

- It has two types of wires, bit wires and element wires. Element wires should be thought as containing/hiding values  $x \in X$ .
- There is a given element wire containing the origin  $x_0$  of the action.
- There are bit gates where both input and output are bit wires.
- There are special element gates which combine bit wires and element wires:

ACTION GATE: It has as inputs some bit wires that encode the group element  $g$  with an hint  $e$ , and an element wire containing  $x$ . Its output is an element wire which contains  $g \star x$ .

EQUALITY GATE: It has two element wires  $x, y$  as inputs, and outputs a bit wire that is 1 if  $x = y$ , and 0 otherwise.

This “type-safe” model can be seen as a possible definition of a *generic group action*; in [MZ22] the authors briefly propose a generic group action framework based on Shoup’s model, stating that it can also model quantum adversaries. Unfortunately, since AGM is incompatible with Shoup’s GGM, for our purposes we decided to follow the type-safe approach.

Notice that, analogously to the group-based TS model, also in our group-action based TS-model, a variant of the *knowledge of exponent assumption* [Dam92] holds by default: by following the wires, for any element wire  $x$  it’s always possible to produce a group element  $g$  such that  $x = g \star x_0$ .

Moreover, we allow the adversary to access the group structure in any case: the hardness of a group action should derive from hiding the group inside the set, and not from the group having a “generic” structure.

Turning now our attention to the multi-party setting, we see that we can define an equivalent of the UC framework, where all the parties are type-safe machines w.r.t. some HEGA  $\mathcal{G}$ . We will usually drop the reference to  $\mathcal{G}$  and assume that the HEGA to which “type-safe” refers can be inferred from the context.

**Definition 7.** Fix an HEGA  $\mathcal{G}$ . A protocol  $\pi$  between parties  $P_1, \dots, P_n$  is said to be type-safe w.r.t.  $\mathcal{G}$  if all  $P_i$  are  $TS_{\mathcal{G}}$  algorithms, and the communication channels consist of both bit wires and element wires. We then say that a type-safe protocol  $\pi$  UC-TS emulates a type-safe protocol  $\phi$  if for any type-safe adversary  $\mathcal{A}$  there is a type-safe simulator  $\mathcal{S}$  such that for any type-safe environment  $\mathcal{Z}$  it holds that  $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{EXEC}_{\phi, \mathcal{S}, \mathcal{Z}}$ .

In this definition, we are using the exact same definitions of the UC framework, but we are restricting machines to be type-safe instead of any generic ITM machine. This means that all communication channels have both bit wires and element wires, and also input/output from/to the environment can be element wires.

We can also define the realization of a type-safe functionality exactly in the same way as in the UC framework, i.e. with the protocol  $\text{IDEAL}_{\mathcal{F}}$  (see Section Section 2.6).

**Definition 8.** Let  $\mathcal{F}$  be a  $TS_{\mathcal{G}}$  functionality. Then the  $TS_{\mathcal{G}}$  protocol  $\text{IDEAL}_{\mathcal{F}}$  is defined as follows:

- There are  $P_1, \dots, P_n$  dummy parties.
- Upon receiving an  $(\text{input}, m, x)$  message (where  $m$  is the content of a bit wire, and  $x$  is the content of an element wire) from the environment, party  $P_i$  forwards  $(m, x)$  to  $\mathcal{F}$ .
- Upon getting a result  $(c, y)$  from  $\mathcal{F}$ , party  $P_i$  sends  $(\text{output}, c, y)$  to the environment.

We say that a  $TS_{\mathcal{G}}$  protocol  $\pi$  UC-TS realizes the functionality  $\mathcal{F}$  if  $\pi$  UC-TS emulates  $\text{IDEAL}_{\mathcal{F}}$ .

Similarly to UC and UC-AGM, we can define the concept of the hybrid execution model and prove the composition theorem. However, we will not prove results in the UC-TS model, but simply use it to compile protocols with additional restrictions, similar to how AGM is interpreted in [Zha22] in relation to the TS/Maurer model.

We can see an example of a two-party TS protocol in Figure 5, where it's also present an ideal hybrid functionality  $\mathcal{F}$ , which is also a TS machine. All communications channels have both bit wires and element wires.

### 3.3 The Algebraic Action Model

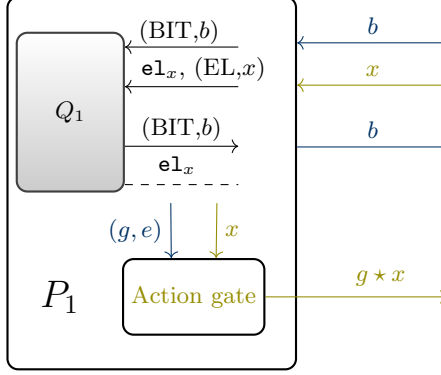
We now introduce the *algebraic action model* (AAM) as a relaxation of the TS model, where adversaries can behave arbitrarily, but must explain the set elements they produce.

First we define what it means for an adversary to behave algebraically.

**Definition 9.** Let  $\mathcal{G}$  be an HEGA, and  $\mathcal{A}$  a PPT algorithm<sup>1</sup>. We say that  $\mathcal{A}$  uses algebraic actions w.r.t.  $\mathcal{G}$  (denoted by  $AA_{\mathcal{G}}$ ) if

- It receives two types of input messages:
  - $(\text{BIT}, b)$ , where  $b$  is a bit-string
  - $(\text{EL}, x)$ , where  $x \in X$  is the representation of some element
- It sends two types of output messages:
  - $(\text{BIT}, b)$ , where  $b$  is a bit-string
  - $(\text{EL}, (g, e, x))$ , where  $g \in G$ ,  $e$  is some hint and  $x$  is one of the previously received  $\text{EL}$  messages.

<sup>1</sup> Remark that we don't impose the use of *element* wires to  $\mathcal{A}$ .



**Figure 6.** Compiled protocol  $AA(\pi, \{Q_1\})$

The meaning of this definition is that if an AA adversary wants to output a set element  $y \in X$ , it must explain it as some  $y = g \star x$  for some  $x \in X$  that it has already seen. Thus, exactly as in the TS/Maurer generic group model adversaries can only employ “generic” algorithms while in the AGM adversaries have access to the group structure and are restricted to “algebraic” algorithms, in our AAM we are also forcing adversaries to behave “algebraically”. In our case this means to produce new elements only via the group action, while being able to access the explicit representation of set elements.

Notice that any given  $TS_G$  algorithm can be translated into an  $AA_G$  algorithm, and thus we can use the AA model as a compiler in the following way.

**Definition 10.** Let  $\pi$  be a  $TS_G$  protocol between parties  $P_1, \dots, P_n$ , and let  $Q = \{Q_{i_1}, \dots, Q_{i_k}\}$  a set of algebraic algorithms, with  $S = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ .

Then  $AA(\pi, Q)$  is another  $TS_G$  protocol where each party  $P_s$  for  $s \in S$  is replaced by  $Q_s$  and its communication wires have been transformed as follows:

- Any incoming bit wire to  $P_s$  gets translated to a  $BIT$  message for  $Q_s$ .
- Any incoming element wire to  $P_s$  gets translated into a  $(EL, x)$  message for  $Q_s$ , and the element wire gets labelled as  $el_x$ .
- Any outgoing  $BIT$  message from  $Q_s$  gets translated into a bit wire from  $P_s$ .
- Any outgoing  $(EL, (g, e, x))$  message from  $Q_s$  gets translated back into an element wire from  $P_s$ , by applying an action gate on  $el_x$  with group element  $(g, e)$ .

The AA compilation on a party  $P_1$  being substituted by an algebraic  $Q_1$  can be seen in Figure 3.3. What this compiler is doing is transforming a fixed subset of parties in a TS protocol into algebraic machines, forcing them to output group elements with which they have generated the element  $y$  they are sending. The goal of this transformation is being able to give a definition of what it means to be “secure in the algebraic action model”. The intuition here is that a protocol  $\pi$  is secure in the AAM model if its compiled version  $AA(\pi)$  is secure in the standard model. This type of approach is exactly the one used by [Zha22] for formally (re)defining security in the AGM.

We conclude the discussion of the AAM with the following informal statement about unsamplable HEGA.

**Proposition 4 (Informal).** Let  $\mathcal{G}$  be an unsamplable HEGA, and let  $\mathcal{A}$  be a PPT algorithm that receives inputs and sends outputs of the form  $(BIT, b)$  and  $(EL, x)$ . Then, if the  $BIT$  inputs give



no information about the group action  $\mathcal{G}$ , there exists an algebraic algorithm  $\mathcal{A}'$  that upon the same inputs of  $\mathcal{A}$  gives the same outputs, giving also an explanation to the EL messages that it outputs.

This informal proposition means that for an unsampleable HEGA, all possible adversaries are actually algebraic; however, this is the closest we can get to a formal result, which cannot exist since it's not clear what it means that BIT messages don't reveal anything about  $\mathcal{G}$ . This is the fundamental issue with the AGM, as highlighted in [KZZ22]; the best resolution of this issue is some kind of type-safe model, but there will always be a gap between the TS model and the standard model.

### 3.4 UC Emulation in the Algebraic Action Model

We now formalize the definition of security in the AA model within the UC framework. First, we use the previous compiler to define the execution of a type-safe protocol against algebraic adversaries.

**Definition 11.** *Let  $\pi$  be a  $TS_{\mathcal{G}}$  protocol, and  $\mathcal{A}, \mathcal{Z}$  be  $AA_{\mathcal{G}}$  algorithms.*

*Denote by  $UC(\pi)$  the  $TS_{\mathcal{G}}$  protocol defined by adding a “dummy” type-safe adversary and environment  $\mathcal{A}', \mathcal{Z}'$ , as in Figure 5. In particular, we have that:*

- *The environment  $\mathcal{Z}'$  can send input messages to the main parties of  $\pi$ , and read their output messages, all of which consist of both a bit wire and an element wire.*
- *The environment  $\mathcal{Z}'$  and the adversary  $\mathcal{A}'$  can communicate freely.*
- *The adversary  $\mathcal{A}'$  has a backdoor channel towards the parties, which also has both bit wires and element wires.*

*We define  $EXEC_{\pi, \mathcal{A}, \mathcal{Z}}(z)$  as the output bit of the environment  $\mathcal{Z}$  in an execution of the protocol  $AA(UC(\pi), \{\mathcal{A}, \mathcal{Z}\})$  with input  $z$ , i.e. we are compiling the “dummy” adversary and environment into meaningful algebraic ones.*

With this definition we are modelling a similar setting of UC-AGM, while using the type-safe model to give a precise definition on the condition “when the adversary outputs a group element” given in UC-AGM [ABK<sup>+</sup>21].

Notice that with our definition we are allowing  $\pi$  to have hybrid type-safe functionalities, and also to have element wires as input/output. For example,  $\mathcal{F}_{RO}$  is a  $TS_{\mathcal{G}}$  algorithm that keeps a list of pairs (element wire, bit-string). When activated with an element wire input, it checks with equality gates if that input is present; if yes, it answers with the bit-string, otherwise it samples a bit-string, answers with it and stores it in the list with the input element wire.

We can finally define what an algebraic emulation of a protocol is, and consequently define an algebraic realization of a type-safe functionality.

**Definition 12.** *Let  $\pi, \phi$  be  $TS_{\mathcal{G}}$  protocols. We say that  $\pi$  UC-AA emulates  $\phi$  if for any  $AA_{\mathcal{G}}$  adversary  $\mathcal{A}$  there is an  $AA_{\mathcal{G}}$  simulator  $\mathcal{S}$  such that for all  $AA_{\mathcal{G}}$  environments  $\mathcal{Z}$  it holds that*

$$EXEC_{\pi, \mathcal{A}, \mathcal{Z}} \approx EXEC_{\phi, \mathcal{S}, \mathcal{Z}}$$

*If  $\mathcal{F}$  is a  $TS_{\mathcal{G}}$  functionality, we say that a  $TS_{\mathcal{G}}$  protocol  $\pi$  UC-AA realizes  $\mathcal{F}$  if  $\pi$  UC-AA emulates  $IDEAL_{\mathcal{F}}$ .*

We can also prove a composition theorem, but, as in UC-AGM, there is an important limitation. In order to obtain emulation, the adversary must give its explanations relative to elements of the same sub-protocol where it's sending the output. Concretely, we say that an  $AA_G$  adversary against a  $TS_G$  composed protocol  $\rho^\pi$  is *non-mixing* if the explanations of messages to be sent to  $\rho$  only use element wires from  $\rho$ , and analogously with  $\pi$ . Therefore, we can state the composition theorem respect to non-mixing adversaries as follow. Note the proof of this theorem is exactly the same as the one given in [ABK<sup>+</sup>21].

**Theorem 5 (Informal).** *Let  $\mathcal{F}_1, \mathcal{F}_2$  be TS functionalities. Let  $\pi$  be a TS protocol that UC-AA realizes  $\mathcal{F}_2$ , and  $\rho$  a protocol that UC-AA realizes  $\mathcal{F}_1$  in the  $\mathcal{F}_2$ -hybrid model. Then protocol  $\rho^\pi$  UC-AA realizes  $\mathcal{F}_1$  against non-mixing algebraic adversaries.*

## 4 The Explicit Isogeny Model

In this section we will introduce the Explicit Isogeny model of computation, which we will then use to prove the security against malicious adversaries of the 2-round OT protocol proposed in [LGd21]. Concretely, the UC-EI model is nothing else than UC-AA instantiated with the action given by CSIDH, where we also incorporate twists for a more accurate model. This means that for any curve  $E$  the adversary must output the secret isogeny path that it used to generate  $E$ .

We will then argue that, given the difficulty of the “hash-to-curve” problem, the CSIDH action is an example of an *unsamplable HEGA*, so that this new model of computation is really close to the standard model.

### 4.1 The CSIDH Action with Twists

Recall that the CSIDH action is  $\star : Cl(\mathcal{O}) \times \mathcal{E} \rightarrow \mathcal{E}$ , where  $\mathcal{O}$  is the order  $\mathbb{Z}[\sqrt{-p}]$  and  $\mathcal{E} = Ell_p(\mathcal{O})$ . However, to fully capture the structure of the  $\mathbb{F}_p$ -rational supersingular isogeny graph it is necessary to also consider twists. This has also been proposed and used by [AEK<sup>+</sup>22] to construct *password authentication key-exchange* (PAKE) protocols and to prove their (in)security. In particular, in this work the authors introduced twists as an external construction, while we will incorporate twists in a slightly larger group action using semidirect products.

We recall that a semidirect product of two groups is given by the following definition.

**Proposition 5 (Semidirect product).** *Let  $H, N$  be groups, and  $\phi : H \rightarrow \text{Aut}(N)$ <sup>2</sup>. Then the cartesian product  $N \times H$  equipped by the operation*

$$(n_1, h_1)(n_2, h_2) = (n_1\phi(h_1)(n_2), h_1h_2)$$

*is a group, denoted by  $N \rtimes_\phi H$ .*

This construction is useful because of the following classical result in the theory of group actions.

**Theorem 6.** *Let  $H$  be a group and  $N$  a  $H$ -group, meaning that there is a map  $\phi : H \rightarrow \text{Aut}(N)$ . Suppose that both  $H$  and  $N$  act on the same set  $X$  in a compatible way, i.e.  $h \cdot (nx) = \phi_h(n) \cdot (hx)$ . Then there is a well-defined action of  $N \rtimes_\phi H$  on the set  $X$  given by  $(n, h) \star x = n \cdot (hx)$ .*

<sup>2</sup>  $\text{Aut}(N)$  denotes the group of all automorphisms of  $N$

We will apply the theorem in our setting, where  $H = \mathbb{Z}/2\mathbb{Z}$ ,  $N = Cl(\mathcal{O})$ ,  $X = \mathcal{E}$ . The action of  $\mathbb{Z}/2\mathbb{Z}$  on  $\mathcal{E}$  is exactly twisting, in particular  $0 \cdot E = E$  and  $1 \cdot E = E^t$ . Moreover,  $\mathbb{Z}/2\mathbb{Z}$  has also a natural action on  $Cl(\mathcal{O})$  given by  $1 \cdot \mathbf{a} = \mathbf{a}^{-1}$ , which corresponds to the map  $\iota : \mathbb{Z}/2\mathbb{Z} \rightarrow \text{Aut}(Cl(\mathcal{O}))$  where  $\iota(1) : \mathbf{a} \mapsto \mathbf{a}^{-1}$ . It is now easy to see that the compatibility requirement is exactly the fact that  $(\mathbf{a} \cdot E)^t = \mathbf{a}^{-1} \cdot E^t$ . We can then construct a new action on  $\mathcal{E}$ , which automatically includes twists in its description.

**Corollary 1.** *There is an action of  $G_{\text{tw}} := Cl(\mathcal{O}) \rtimes_{\iota} \mathbb{Z}/2\mathbb{Z}$  on  $\mathcal{E}$  given by  $(\mathbf{a}, 0) \star E = \mathbf{a} \cdot E$  and  $(\mathbf{a}, 1) \star E = \mathbf{a} \cdot E^t$ .*

Since  $|G_{\text{tw}}| = 2|Cl(\mathcal{O})|$ , the action will not be regular anymore; indeed, for any  $E = \mathbf{a}E_0$  it's easy to see that its stabilizer is  $Stab(E) = \{(1, 0), (\mathbf{a}^2, 1)\}$ .

We conclude by showing that what we have constructed is still an HEGA.

**Proposition 6.** *Let  $\star : G_{\text{tw}} \times \mathcal{E} \rightarrow \mathcal{E}$  the action defined above. Then  $\mathcal{G}_{\text{tw}} = (G_{\text{tw}}, \mathcal{E}, \star, \mathcal{D}_{G_{\text{tw}}}, E_0)$  such that:*

- $E_0$  is the same as the CSIDH action, namely the curve  $y^2 = x^3 + x$
- $\mathcal{D}_{G_{\text{tw}}}$  is defined by independently sampling  $(\mathbf{a}, e)$  from  $Cl(\mathcal{O})$  as in CSIDH, and choosing a random bit  $b \in \mathbb{Z}/2\mathbb{Z}$ ; the group element is  $(\mathbf{a}, b)$  and the hint is  $e' = (e, b)$

*is an HEGA. Moreover,  $\mathcal{D}_{G_{\text{tw}}}$  is close to the uniform distribution.*

*Proof.* The last claim follows directly from Lemma 1, i.e. that the CSIDH sampling is close to the uniform distribution on  $Cl(\mathcal{O})$ ; since we then uniformly choose a bit, we can see that  $\mathcal{D}_{G_{\text{tw}}}$  approximates the uniform distribution on  $Cl(\mathcal{O}) \times \mathbb{Z}/2\mathbb{Z}$ .

It is also trivial to show that we can evaluate the action given the hints. Indeed, the hint  $e' = (e, b)$  tells us if we have to twist or not with  $b$ , and then we can apply the CSIDH action algorithm with the secret exponent vector  $e$ .

The other properties, and in particular those on the set  $\mathcal{E}$ , follow from CSIDH. ■

## 4.2 The UC-EI Model

We are now ready to define the Explicit Isogeny model, which is an instantiation of UC-AA with  $\mathcal{G}_{\text{tw}}$ . More concretely, we see that our compiler turns the adversaries into EI-adversaries.

**Definition 13.** *Let  $\mathcal{G}_{\text{tw}}$  be the HEGA defined before. We say that an algorithm  $\mathcal{A}$  uses Explicit Isogenies (EI) if its communication tapes have messages of the type  $(\text{bit}, m)$  and  $(\text{curve}, E)$ , where  $m$  is a bit-string and  $E$  is an element of  $\mathcal{E}$ , i.e., a supersingular curve over  $\mathbb{F}_p$ .*

*Moreover, for any outgoing message  $(\text{curve}, E)$ ,  $\mathcal{A}$  must also send an explanation  $(\mathbf{a}, e, E')$  such that  $E = \mathbf{a} \star E'$ , where  $E'$  is one of the previous incoming curve messages or its twist.*

Notice that for our UC-AA definition to make sense, it's necessary that the original protocol is type-safe w.r.t.  $\mathcal{G}_{\text{tw}}$ , so incorporating twists into the action is a fundamental step of the construction, otherwise we couldn't describe protocols that use twists, such as the one proposed by Lai et al. [LGd21].

We also restate our definition of UC emulation, which is exactly that given for UC-AA for the specific case of  $\mathcal{G}_{\text{tw}}$  protocols against EI adversaries.

**Definition 14.** Given two  $TS_{\mathcal{G}_{\text{tw}}}$  protocols  $\pi$  and  $\phi$ , we say that  $\pi$  UC-EI emulates  $\phi$  if for any efficient EI-adversary  $\mathcal{A}$  there is an efficient EI-simulator  $\mathcal{S}$  such that for any efficient EI-environment  $\mathcal{Z}$  we have that

$$\text{EXEC}_{\phi, \mathcal{S}, \mathcal{Z}} \approx \text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}.$$

A  $TS_{\mathcal{G}_{\text{tw}}}$  protocol  $\pi$  is said to UC-EI realise an ideal functionality  $\mathcal{F}$  if  $\pi$  UC-EI emulates  $\text{IDEAL}_{\mathcal{F}}$ .

Like UC-AGM, also UC-EI limits the capabilities of the adversary, so it may seem that it's less expressive than the plain model of UC security. However, as we will discuss in the next sections, there is a strong evidence suggesting that any PPT adversary  $\mathcal{A}$  behaves like in the explicit isogeny model, in particular given the fact that “hashing” to a supersingular curve seems to be very hard.

### 4.3 The EI Model and the Sampling Problem

One of the main open problems of isogeny-based cryptography is how to sample a supersingular curve without taking a random isogeny walk from another known curve; this is also called the “hashing-to-curve” problem, which can be roughly stated as follows.

*Problem 1 (Informal).* Find an efficient sampling algorithm  $E \leftarrow_{\$} SS_p^3$ , from which computing  $\text{End}(E)$  is still hard.

More concretely, the sampling problem is defined and studied in [MMP22], where it is called the cSRS (cryptographic Supersingular Random Sampling) problem. In this work, the authors review some known methods to generate supersingular elliptic curves and propose some possible ideas for new algorithms, which still have exponential complexity.

The main method of constructing supersingular elliptic curves is due to Bröker [Bro09], via reduction of curves over number fields that have complex multiplication (CM). However, in [CPV20] the authors show that for the special curves generated by Bröker’s algorithm it is possible to efficiently find an isogeny to a base curve. In particular, they work in the CSIDH setting and find a smooth ideal connecting the CM curve to  $E_0 : y^2 = x^3 + x$ .

In another very similar work [BBD<sup>+</sup>22], the authors try to find efficient ways to hash into the isogeny graph, but in the end they conclude that their ideas are still not enough to solve the problem.

Given all the failed attempts, we could introduce a new hardness assumption based on this problem.

**Assumption 1** For any PPT algorithm  $\mathcal{A}$  that receives as input a prime number  $p$  and outputs a supersingular  $j$ -invariant  $j \in \mathbb{F}_{p^2}$ , there exists a PPT algorithm  $\mathcal{A}'$  that outputs the pair  $(j, R)$ , where  $R = \text{End}(E)$  is the endomorphism ring of a curve  $E$  with  $j(E) = j$ .

We can also use a slightly different variant of the problem, that only deals with isogeny paths and not endomorphism rings.

**Assumption 2** Sampling a supersingular curve  $E/\mathbb{F}_{p^2}$  without learning a path from a known curve is hard. More precisely, for any PPT algorithm  $\mathcal{A}(p, j_0, j_1, \dots, j_n)$ , that outputs a supersingular  $j$ -invariant  $j \in \mathbb{F}_{p^2}$  knowing a list of some  $j$ -invariants, there exists a PPT algorithm  $\mathcal{A}'$  that outputs a computable isogeny  $\phi : E_i \rightarrow E$ , where  $j(E) = j$  and  $j(E_i) = j_i$ .

<sup>3</sup> We denote by  $SS_p$  the set of supersingular  $j$ -invariants over  $\mathbb{F}_p^2$ .

It is important to notice that the two different Assumptions 1 and 2 are not equivalent: knowing an isogeny  $\phi : E_0 \rightarrow E$  is equivalent to knowing  $\text{End}(E)$  only if we can compute  $\text{End}(E_0)$  (for example if it is  $y^2 = x^3 + x$ ), but in an interactive protocol a party can receive a supersingular curve from other parties, without being able to know its endomorphism ring.

The assumption is thus trying to model exactly the multi-party computation setting, by forcing any party to generate new curves only by walking in the isogeny graph, starting from curves that have been sent to it.

Moreover, in [BBD<sup>+</sup>22] the authors highlight some variants for the hashing problem, in particular the problem of sampling  $\mathbb{F}_p$ -rational supersingular curves. It is related to the general problem, and it likely seems as difficult, but it is very hard to prove an equivalence between them. There are also no better algorithms to hash into the  $\mathbb{F}_p$ -graph than those that hash into the  $\mathbb{F}_{p^2}$ -graph.

We will then pose another assumption, specific to the CSIDH setting.

**Assumption 3** *Sampling a supersingular curve  $E/\mathbb{F}_p$  without learning a path from a known curve is hard.*

*More precisely, for any PPT algorithm  $\mathcal{A}(p, E_0, E_1, \dots, E_n)$  that outputs a supersingular curve  $E/\mathbb{F}_p$  knowing a list of some supersingular curves  $E_i/\mathbb{F}_p$ , there exists a PPT algorithm  $\mathcal{A}'$  that outputs a computable  $\mathbb{F}_p$ -rational isogeny  $\phi : E'_i \rightarrow E$ , where  $E'_i$  is  $E_i$  or one of its twists.*

Notice that this is a quite literal translation of Assumption 2 into the  $\mathbb{F}_p$ -isogeny graph, where we have to use curves instead of  $j$ -invariants: any supersingular invariant  $j \in \mathbb{F}_p$  will correspond to multiple curves which are  $\overline{\mathbb{F}}_p$ -isomorphic, but not  $\mathbb{F}_p$ -isomorphic, i.e. all the twists. Moreover, this assumption also captures the fact that twists are easy to compute, so an algorithm can easily generate  $E^t$  from  $E$ , even without knowing an isogeny from  $E$  to  $E^t$ .

This new assumption is actually needed, and it doesn't seem to follow directly from the other two. Indeed, we know that the CSIDH problem and the `EndRing` problem are equivalent, but this is insufficient to relate Assumption 1 and Assumption 3 for the exact same reason for which Assumption 1 and Assumption 2 are not equivalent.

Trying to relate Assumptions 2 and 3 means translating an  $\mathbb{F}_{p^2}$ -isogeny  $\phi : E_1 \rightarrow E_2$  into a smooth ideal. If the full  $\text{End}(E_1)$  was known, we could use all the results from Section 2.5 to get a smooth element of  $Cl(\mathcal{O})$ ; but in general we cannot hope to know  $\text{End}(E_1)$ , which is the case when  $E_1$  is given to us from another party. The problem of translating  $\mathbb{F}_{p^2}$  isogenies into  $\mathbb{F}_p$  isogenies seems to be a very interesting one, but for now it still remains open.

We conclude the section by describing what those assumptions mean for the EI model, starting from the following immediate result.

**Proposition 7.** *Under Assumption 3,  $\mathcal{G}_{\text{tw}}$  is an unsampleable HEGA.*

In conjunction with Proposition 4, this means that our UC-EI model does not impose any actual restrictions, and can almost be thought as equivalent to the plain UC model, with all the caveats of that informal proposition.

Notice that if instead the assumption doesn't hold and parties have other ways to generate supersingular curves, then the EI model is actually less expressive than the plain model, in the same way that the AGM denies the sampling of random group elements, while it is a very easy operation both for finite fields and for elliptic curves.

## 5 Actively-Secure 2-Round Isogeny-Based OT

In this section we describe a 2-round OT protocol secure against malicious adversaries, give its security proof in the UC-EI model and finally compare it with other low round isogeny-based protocols. We describe in Section 5.4 a variant of this protocol with 3 rounds of complexity, but without needing a trusted setup.

### 5.1 The Twist OT Protocol

The protocol  $\Pi_{\text{tw}}$ , described in Figure 7, is exactly the 2-round protocol proposed by Lai et al. [LGd21].

Let  $(\text{KeyGen}, \text{Enc}, \text{Dec})$  be an IND-CPA symmetric encryption scheme and  $H : \mathcal{E} \rightarrow \mathcal{K}$  be modelled as a type-safe random oracle; the protocol works as follows. Given a trusted setup curve  $E$ , the sender  $P_S$  and the receiver  $P_R$  independently sample an ideal in the ideal class group  $Cl$ , i.e.,  $P_S$  samples  $s \leftarrow_{\$} Cl$  and computes  $A = s \star E$  whereas  $P_R$  samples  $r \leftarrow_{\$} Cl$  and computes  $C = r \star E$ . Now, if the receiver's choice bit  $\sigma$  is 0,  $P_R$  computes  $C_\sigma$  as  $C = r \star E$ , otherwise as  $C^t$ . Notice that, by Lemma 2, the curve  $C_\sigma$  statistically hides the choice bit. Receiving  $C_\sigma$ ,  $P_S$  computes both  $k_0 = H(s \star C_\sigma)$  and  $k_1 = H(s \star C_\sigma^t)$ . In this way, when  $\sigma = 0$ ,  $k_0 = H(r \star A)$  and  $k_1 = (s \star C^t)$ , otherwise if  $\sigma = 1$ , then  $k_0 = H(s \star C^t)$  and  $k_1 = H(r \star A)$ . Similarly to what happens in the discrete-logarithm setting, when  $P_R$  receives  $A$  and  $c_b = \text{Enc}_{k_b}(m_b)$ ,  $b \in \{0, 1\}$ , it can recover  $k_\sigma$  by  $H(r \star A)$  and decrypt the corresponding message  $m_\sigma$ .

Notice that this whole protocol is actually type-safe w.r.t.  $\mathcal{G}_{\text{tw}}$ , since the only operations we do on curves are twists and CSIDH actions.

The functionality  $\mathcal{F}_{\text{TSC}}$  is also type-safe, and is defined by sampling  $(t, e)$  from  $\mathcal{D}_{\mathcal{G}_{\text{tw}}}$ , and then using an action gate on the element wire  $x_0$  to get an element wire corresponding to  $t \star x_0$ , which will be its output to any party.

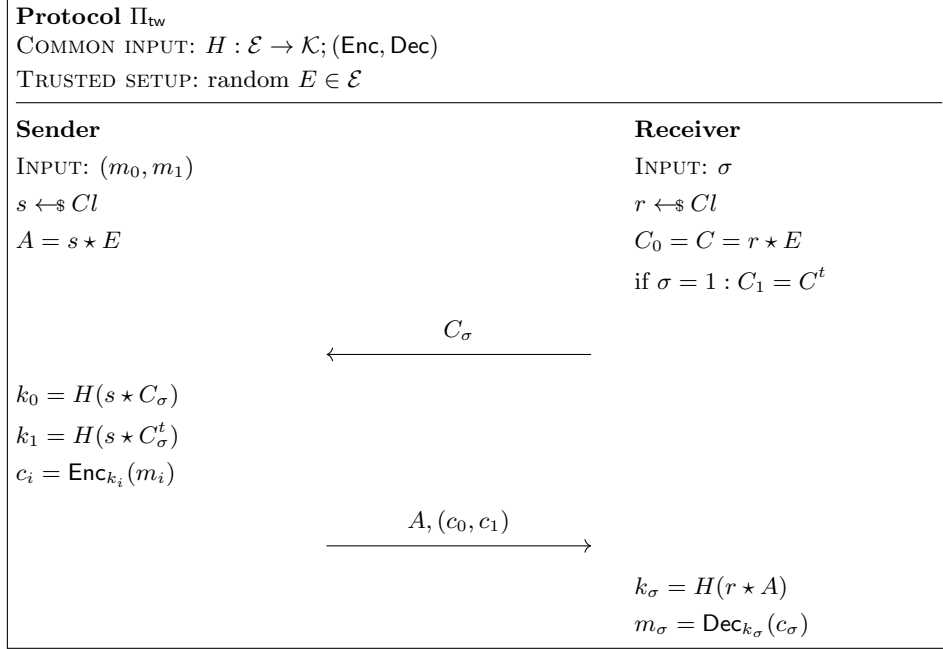
This protocol was only proved to be semi-honest secure in the UC framework by Lai et al.; in the same paper, the authors give a maliciously-secure version of it that requires additional two rounds of communication to permit the extraction of the input of a malicious receiver in the UC proof. We now show that this is not needed in our UC-EI setting.

**Theorem 7.** *The protocol  $\Pi_{\text{tw}}$ , described in Figure 7, UC-EI realizes the functionality  $\mathcal{F}_{\text{OT}}$  (Figure 2) in the  $(\mathcal{F}_{\text{RO}}, \mathcal{F}_{\text{TSC}})$ -hybrid model in the presence of malicious adversaries and static corruptions, if the encryption scheme  $(\text{KeyGen}, \text{Enc}, \text{Dec})$  is IND-CPA secure and the CSIDH vectorization problem is hard.*

*Proof.* We distinguish the main two cases of honest  $P_S$  and corrupt  $P_R$  and corrupt  $P_S$  and honest  $P_R$ . Proving security in the remaining two cases is straightforward.

**HONEST SENDER AND CORRUPT RECEIVER.** We first describe the simulator  $\mathcal{S}$ . Recall, that in order to emulate the adversary  $\mathcal{A}$ ,  $\mathcal{S}$  has to extract the input of the corrupt receiver to forward it to the OT functionality.

*Simulation.* Throughout the execution,  $\mathcal{S}$  simulates the random oracle  $H$  by answering every new query with a random value from the relevant set and maintaining a list of past queries to answer repeated queries consistently. More concretely,  $\mathcal{S}$  keeps a list  $L$  in  $\mathcal{E} \times \mathcal{K}$  in which it stores all the past queries. It initializes the random oracle with an empty list, then for each query  $X \in \mathcal{E}$  it checks whether  $(X, k) \in L$ : if this is the case, returns  $k$ , otherwise it samples a random  $k \leftarrow_{\$} \mathcal{K}$ , adds  $(X, k)$  to  $L$  and returns  $k'$ . The simulator is defined by the following instructions:



**Figure 7.** The twist OT protocol by Lai et al.

- Emulate the trusted setup step, defining the curve  $E = t \star E_0$ , with a randomly sampled  $t \in Cl(\mathcal{O})$ .
- Set its public key as the honest sender  $A = s \star E$ , for a random  $s \leftarrow_{\$} Cl(\mathcal{O})$ .
- When receiving the curve  $C$  from the adversary, also obtain an explanation  $C = x \star E_R$ , with  $E_R = E_0, E$  or  $E^t$ . If  $E_R = E$  set  $\sigma = 0$ , if  $E_R = E^t$  set  $\sigma = 1$ , otherwise set  $\sigma = -1$ . If  $\sigma \neq -1$ , query the functionality and get the message  $m_\sigma$ .
- Sample two random keys  $k_i \leftarrow \text{KeyGen}()$  and for any additional query to  $H$  proceed as follows .
  - If query is  $s \star C$ : if  $\sigma \neq 0$  send **abort** to the ideal functionality, otherwise returns  $k_0$ ;
  - If query is  $s \star C^t$ : if  $\sigma \neq 1$  send **abort** to  $\mathcal{F}_{OT}$ , otherwise returns  $k_1$ .
- Set any  $m_i$  that it doesn't know to 0 (i.e.  $m_{1-\sigma}$  if  $\sigma \in \{0, 1\}$ , both  $m_0, m_1$  otherwise); then it computes  $c_i = \text{Enc}_{k_i}(m_i)$ .
- Finally, send  $A, c_0, c_1$  to the adversary.

*Indistinguishability.* We now prove indistinguishability between the real and ideal execution. Let  $\mathcal{Z}_S$  denote  $\text{EXEC}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ , while  $\mathcal{Z}_\pi$  denote  $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$ . Let **Ab** be the event that  $\mathcal{S}$  aborts in an ideal execution. Write

$$s = \Pr[\mathcal{Z}_S = 1 \mid \text{Ab}], \quad s' = \Pr[\mathcal{Z}_S = 1 \mid \neg \text{Ab}], \quad p = \Pr[\mathcal{Z}_\pi = 1] \quad a = \Pr[\text{Ab}].$$

Then we have

$$\begin{aligned} & \left| \Pr[\mathcal{Z}_S = 1] - \Pr[\mathcal{Z}_\pi = 1] \right| = \left| sa + s'(1 - a) - p \right| = \left| a(s - s') + s' - p \right| \\ & \leq 2a + \left| s' - p \right| = 2\Pr[\text{Ab}] + \left| \Pr[\mathcal{Z}_S = 1 \mid \neg \text{Ab}] - \Pr[\mathcal{Z}_\pi = 1] \right|. \end{aligned}$$

The theorem will then follow from the fact that both quantities are negligible. Indeed if  $\mathcal{S}$  aborts, then we can solve a vectorization CSIDH problem, while if  $\mathcal{Z}$  can distinguish we can break the

	$y \star E_0$	$y \star E$	$y \star E^t$	$y \star A$	$y \star A^t$
$C = x \star E_0, s \star C$	$s = yx^{-1}$	$s = ytx^{-1}$	$s = yt^{-1}x^{-1}$	$t = xy^{-1}$	$s^2 = x^{-1}t^{-1}$
$C = x \star E_0, s \star C^t$	$s = yx$	$s = ytx$	$s = yt^{-1}x$	$t = x^{-1}y^{-1}$	$s^2 = xt^{-1}$
$C = x \star E, s \star C^t$	$s = xyt$	$s = xyt^2$	$s = xy$	$t = x^{-1}y^{-1}$	$s^2 = xy$
$C = x \star E^t, s \star C$	$s = x^{-1}yt$	$s = x^{-1}yt^2$	$s = x^{-1}y$	$t = xy^{-1}$	$s^2 = x^{-1}y$

**Table 1.** The computable solutions to the CSIDH problem

IND-CPA property of the encryption scheme. More concretely, suppose that  $\mathcal{S}$  does not abort, then we can construct an adversary  $\mathcal{D}$  for the IND-CPA game as we describe in what follows.  $\mathcal{D}$  internally runs  $\mathcal{Z}$  against  $\mathcal{S}$ , but stopping the execution before the simulator computes  $c_{1-\sigma}$ .

Then  $\mathcal{D}$  takes the input  $(m_0, m_1)$  for the honest sender, and sends to the IND-CPA oracle the pair of messages  $(0, m_{1-\sigma})$ , which returns a ciphertext  $c$ . At this point  $\mathcal{D}$  resumes the execution, but it sets  $c_{1-\sigma} = c$ . Finally  $\mathcal{D}$  outputs whatever  $\mathcal{Z}$  outputs. Notice that when the bit  $b$  of the IND-CPA oracle is 1,  $\mathcal{D}$  runs a perfect emulation of the real protocol, while if  $b = 0$ ,  $\mathcal{D}$  is running  $\mathcal{S}$ . This means that

$$\begin{aligned} \text{Adv}_{\mathcal{D}, \mathcal{E}}^{\text{ind-cpa}} &= \left| \Pr[\mathcal{D} = 1 \mid b = 0] - \Pr[\mathcal{D} = 1 \mid b = 1] \right| \\ &= \left| \Pr[\mathcal{Z}_{\mathcal{S}} = 1 \mid \neg \text{Ab}] - \Pr[\mathcal{Z}_{\pi} = 1] \right|. \end{aligned}$$

In particular  $\mathcal{Z}$  cannot distinguish  $\mathcal{S}$  and the real world if the encryption scheme  $\mathcal{E}$  is IND-CPA, in the case that  $\mathcal{S}$  doesn't abort.

We now estimate the probability of  $\mathcal{S}$  aborting. Suppose then that we have a CSIDH problem  $E_1 = a \star E_0$  we want to solve. We create two possible solvers for this problem:

- Algorithm  $\mathcal{D}_1$  will run  $\mathcal{S}$  with  $E_1$  as trusted setup. Then it will check if it can compute  $a$  from the queries and explanations that  $\mathcal{Z}$  makes to the random oracle.
- Algorithm  $\mathcal{D}_2$  will run  $\mathcal{S}$  with  $b \star E_0$  as trusted setup and  $b \star E_1$  as sender's public key. Then it will check queries to compute a value  $a'$  such that  $a' \star (b \star E_0) = b \star E_1$ , which means  $a' = a$ .

In Table 1, we show how  $\mathcal{D}_1$  and  $\mathcal{D}_2$  can compute the solutions from the query. The rows are indexed by the explanation of the curve  $C$  and the query, while the columns are the explanation of the query. We now compute the probability that the simulator aborts,  $\Pr[\mathcal{S} \text{ aborts}]$ , and estimate it with the advantages of  $\mathcal{D}_1, \mathcal{D}_2$  for the CSIDH problem.

Let  $T_1$  be the event that  $\mathcal{Z}$  makes one of the “forbidden” queries and explains it as  $y \star A$  (so that  $t$  can be computed); let  $T_2$  be the event that a forbidden query is made and is explained differently from  $y \star A$  (in which case  $s$  can be computed).

Notice that  $\mathcal{S}$  only aborts when a forbidden query is made, so we have that  $\Pr[\mathcal{S} \text{ aborts}] = \Pr[T_1] + \Pr[T_2]$ . Moreover  $\mathcal{D}_i$  wins with probability 1 if event  $T_i$  happens, so we have that

$$\text{Adv}_{\mathcal{D}_i}^{\text{csidh}} \geq 1 \cdot \Pr[T_i] + \frac{1}{\#\text{Cl}(\mathcal{O})} \Pr[\neg T_i] \geq \Pr[T_i].$$

In particular, we get that  $\Pr[\mathcal{S} \text{ aborts}] \leq \text{Adv}_{\mathcal{D}_0}^{\text{csidh}} + \text{Adv}_{\mathcal{D}_1}^{\text{csidh}}$ , from which we can finally conclude

$$\left| \Pr[\mathcal{Z}_{\mathcal{S}} = 1] - \Pr[\mathcal{Z}_{\pi} = 1] \right| \leq \text{Adv}_{\mathcal{D}}^{\text{ind-cpa}} + \text{Adv}_{\mathcal{D}_0}^{\text{csidh}} + \text{Adv}_{\mathcal{D}_1}^{\text{csidh}},$$



which proves indistinguishability, provided that the encryption scheme is IND-CPA and the CSIDH problem is hard.

**CORRUPT SENDER AND HONEST RECEIVER.** As in the previous case, we first describe the simulator and then we argue indistinguishability between the real and ideal execution.

*Simulation.* The simulator  $\mathcal{S}$  handles random oracles queries as in the previous case and does the following.

- Backdoor the trusted setup as before: sample  $t \leftarrow_{\$} Cl$  and set  $E = t \star E_0$
- Sample  $r \leftarrow_{\$} Cl$  and compute  $C = r \star E$ . Set  $\sigma = 0$  and send  $C$  to  $\mathcal{A}$ . Then proceed as an honest party would do.
- If, at some point,  $\mathcal{A}$  sends **abort**, then forward **abort** to the OT functionality.
- If receive  $(A, c_0, c_1)$ , together with an explanation for  $A$ , from  $\mathcal{A}$ , using  $t$  can recover both the keys  $k_0$  and  $k_1$  as follows:  $k_0 = H(r \star A)$  and  $k_1 = H(r^{-1}t^{-2} \star A)$ ; then decrypt the two messages with the computed keys  $m_i = \text{Dec}_{k_i}(c_i)$ . Send  $m_0, m_1$  to the functionality.
- Output whatever the adversary outputs and halt.

*Indistinguishability.* In the trusted setup, the simulator backdoors the public curve, but this is unnoticeable to the adversary. After the setup phase, in the real protocol the curve  $C$  sent by the receiver is either  $r \star E$  or  $(r \star E)^t$  depending on whether  $\sigma = 0$  or  $\sigma = 1$ , respectively. In the former case the messages received by  $\mathcal{A}$  are identically distributed in the two executions, in the latter case, by Lemma 2, the messages are statistically close. Finally, using its knowledge of  $t$  it is straightforward to see that  $\mathcal{S}$  is able to correctly extract the input of  $\mathcal{A}$ . Therefore we can conclude that the two executions are indistinguishable. ■

## 5.2 Other OT Protocols

We can apply our model to prove UC-EI security of other isogeny-based OT protocols. We decided to only show this for a variant of  $\Pi_{\text{tw}}$  that does not require a trusted setup, but unfortunately needs 3 rounds of communication. The protocol  $\Pi_{\text{tw}^3}$ , described in Figure 9, is similar to  $\Pi_{\text{tw}}$ , except that now it is the sender  $P_S$  that generates the curve  $E$  and sends it to the receiver  $P_R$ .

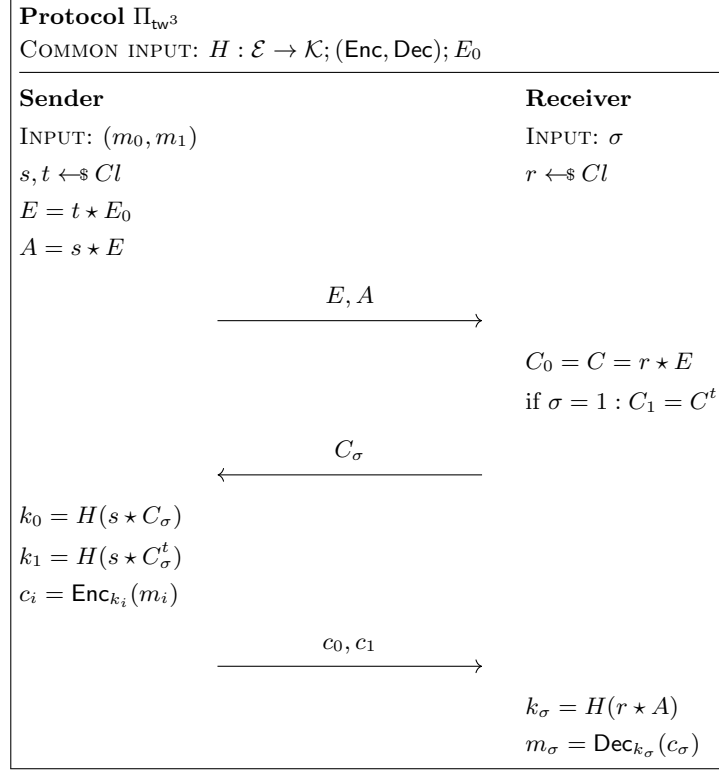
We can prove the following result.

**Theorem 8.** *The protocol  $\Pi_{\text{tw}^3}$ , described in Figure 9, CMEI-realizes the functionality  $\mathcal{F}_{\text{OT}}$  in the random oracle model if the encryption scheme (KeyGen, Enc, Dec) is IND-CPA secure.*

*Proof (Sketch).* As before we distinguish between the two main cases of corrupt sender and honest receiver and honest sender and corrupt receiver.

**CORRUPT SENDER AND HONEST RECEIVER.** As in the other proof, the twisting statistically hides the choice bit, since the lemma holds for any starting curve, even possibly ones that have been maliciously generated.

**HONEST SENDER AND CORRUPT RECEIVER.** This also works as in the other proof, since by simulating the honest sender we know  $t$  such that  $E = t \star E_0$ , which otherwise we knew by simulating the trusted setup functionality. ■



**Figure 8.** The twist protocol without trusted setup

### 5.3 Efficiency and Comparison

In Table 5.3, we give a detailed comparison between the protocol  $\Pi_{\text{tw}}$  presented in Section 5.1 and other isogeny-based OT protocols that have been proposed in the last few years. Notice that all the SIDH-based schemes are not secure any more due to the recent attacks to the CSSI problem (Problem 2 in Section 2.3).

In the table, we report the number of rounds  $\#R$ , the number of isogenies computed by the sender and the receiver, respectively,  $\#(\text{PK}_S, \text{PK}_R)$ , the power of the adversary, the proof framework, the model of computation, and the underlying hardness assumption on which the security of the protocol is based on. The broken assumptions are highlighted in red. We denote the security parameter by  $\lambda$ .

A brief description of some of the works cited in Table 5.3 was already given in [LGd21]. We recall here their main properties for completeness. In the first rows, we have semi-honest secure protocols. The first one is the paper by Barreto et al. [BOB18], which introduces a protocol based on a variant of SIDH. More precisely, it constructs a SIDH public key and creates a second “fake” public key by perturbing the public torsion points. Based on this indistinguishability assumption, Branco et al. [BDGM19] build an instantiation of their proposed OT protocol framework. This latter protocol achieves malicious UC-security, but requires 4 rounds of communication.

<sup>4</sup> The assumption holds for “generic” group actions, but is trivially broken for CSIDH, which is the only post-quantum (R)EGA we know of.

Reference	# R	# (PK <sub>S</sub> , PK <sub>R</sub> )	Adversary	Proof	Model	Assumption
[BOB18]	3	3, 2	Semi-honest	Plain	ROM+CRS	Variant of SSDDH
[dOPS20] I	2	3, 2	Semi-honest	UC	ROM	ParallelEither
[dOPS20] II	3	5, 2	Semi-honest	UC	ROM	ParallelBoth <sup>4</sup>
[LGd21] I	2	3, 2	Semi-honest	UC	ROM+TSC	Inv-CSIDH
[BDGM19]	4	4, 2	Malicious	UC	ROM	Variant of SSDDH
[Vit19] I	3	4, 2	Malicious	Game	ROM	2-inv-DDHP <sup>4</sup>
[Vit19] II	3	4, 2	Malicious	Game	ROM	2-inv-CSSI
[ADMP20]	2	4λ, λ + 3	Malicious	SSP	Plain	wPR-EGA
[ADMP20] + [PVW08]	2	4λ, λ	Malicious	UC	CRS	wPR-EGA
[dOPS20] + [DGH <sup>+</sup> 20]	2	poly(λ)	Malicious	UC	ROM+TSC	ParallelDouble
[LGd21] II	4	5, 6	Malicious	UC	ROM+TSC	Rec-CSIDH
[BMM <sup>+</sup> 22] I	2	O(λ), O(λ)	Malicious	UC	ROM+CRS	Comp-CSIDH
[BMM <sup>+</sup> 22] II	4	O(λ), O(λ)	Malicious	Simulation	Plain	Comp-CSIDH
Π <sub>tw</sub> ([LGd21] + <b>this work</b> )	2	3, 2	Malicious	UC-EI	ROM+TSC	Vec-CSIDH
Π <sub>tw</sub> <sup>3</sup> <b>this work</b>	3	4, 2	Malicious	UC-EI	ROM	Vec-CSIDH

**Table 2.** Comparison of some properties of proposed OT protocols

In [dOPS20], the authors introduce the concept of *masking*, which generalizes the one of hard homogeneous spaces. This allows them to create masks from both SIDH and CSIDH. The paper contains two passively secure OT protocols, one with two rounds, derived from the Shamir-3-Pass key transportation scheme, and the other with three rounds derived from the CO protocol. In addition, the authors prove that their two-round protocol can be extended to be secure against malicious adversaries using a transformation by Döttling et al. [DGH<sup>+</sup>20], which increases the complexity of the protocol as a side effect. As mentioned before, the protocols are based on masking assumptions, ParallelEither, ParallelBoth and ParallelDouble, that can be instantiated with isogeny-based assumptions. We refer to [dOPS20] for additional details.

Another malicious secure scheme is given by Vitse in [Vit19]. Here the author constructs an exponentiation-only protocol, that can be instantiated both with CSIDH and SIDH, in the latter case using dual isogenies. In Diffie-Hellman terms, this means that the receiver gets the public keys  $g^{a_0}, g^{a_1}$ , sends back  $(g^{a_\sigma})^b$  and receives  $g^{a_\sigma \cdot b \cdot a_i^{-1}}$  as possible keys. The resulting OT protocol has three rounds, and is proved to be secure against malicious adversaries but using a game-based definition of security for OT. This protocol is almost the same of the three-round protocol by [dOPS20]. The newly introduced hardness assumption 2-inv-DDHP is analogue to ParallelBoth, but unfortunately they do not hold when instantiated with CSIDH due to the existence of twists, as seen in [Fel19].

In [ADMP20], the authors introduce a new framework based on group actions, from which they derive new cryptographic primitives based on CSIDH. In particular, they construct a “dual-mode encryption scheme” which allows them to use the framework by Peikert et al. [PVW08] to produce an OT protocol that is UC-secure against malicious adversaries. The resulting protocol has only two rounds, but given a security parameter  $\lambda$ , it needs to generate  $O(\lambda)$  public keys and compute  $O(\lambda)$  isogenies. They also directly build a statistically sender-private OT protocol, which still needs  $O(\lambda)$  public key operations. The security of these protocols is related to a variant of EGA that we have introduced in Section 2.

The recent paper [BMM<sup>+</sup>22] constructs a 4-round maliciously secure protocol in the plain model, and a 2-round UC-secure protocol in the ROM+CRS model. Both protocols are based on the computational CSIDH problem, but need a number of isogeny computation that is linear in the security parameter  $\lambda$ .

Lai et al. [LGd21] describe the 2-round semi-honest protocol that we prove to be maliciously secure in the UC-EI model. This protocol is particularly efficient compared to other isogeny-based protocols as it requires only 2 round of communications and a constant, very low number of isogeny computations. The protocol requires both a random oracle and a trusted setup.

It is possible to remove the trusted setup assumption, as we show in Section 5.4; however, this comes at the cost of increasing the number of rounds to 3.

In conclusion, by proving UC-EI security for the protocol  $\Pi_{\text{tw}}$ , we can claim security against malicious adversaries of a two-round OT protocol that only requires 3 isogeny computation from the sender side and 2 isogeny computation for the receiver. Removing one of the two assumptions, i.e., either ROM or TSC, without compromising the efficiency of the protocol, remains a fascinating open question, whose answer will probably require a completely different approach.

#### 5.4 Other OT Protocols

We can apply our model to prove UC-EI security of other isogeny-based OT protocols. We decided to only show this for a variant of  $\Pi_{\text{tw}}$  that does not require a trusted setup, but unfortunately needs 3 rounds of communication. The protocol  $\Pi_{\text{tw}^3}$ , described in Figure 9, is similar to  $\Pi_{\text{tw}}$ , except that now it is the sender  $P_S$  that generates the curve  $E$  and sends it to the receiver  $P_R$ .

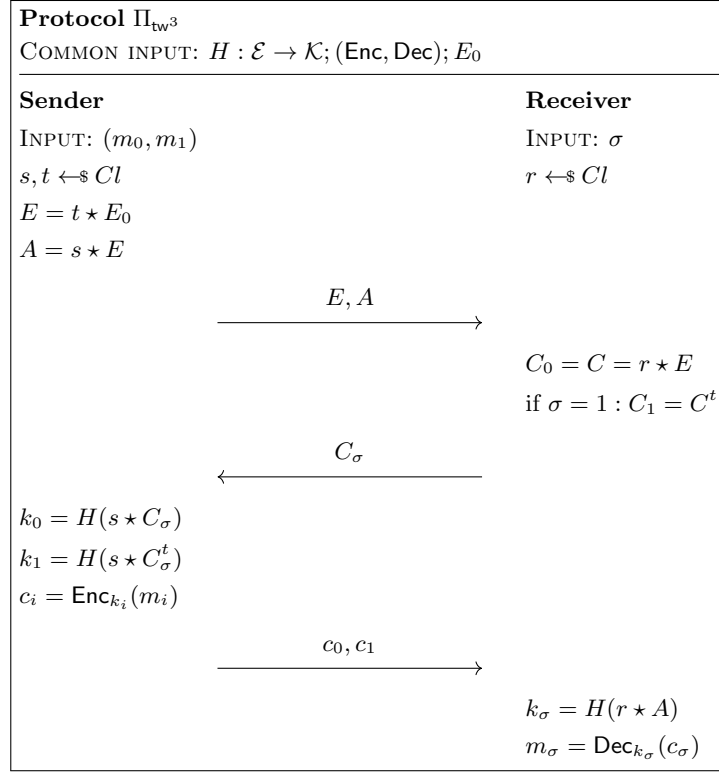
We can prove the following result.

**Theorem 9.** *The protocol  $\Pi_{\text{tw}^3}$ , described in Figure 9, CMEI-realizes the functionality  $\mathcal{F}_{\text{OT}}$  in the random oracle model if the encryption scheme (KeyGen, Enc, Dec) is IND-CPA secure.*

*Proof (Sketch).* As before we distinguish between the two main cases of corrupt sender and honest receiver and honest sender and corrupt receiver.

**CORRUPT SENDER AND HONEST RECEIVER.** As in the other proof, the twisting statistically hides the choice bit, since the lemma holds for any starting curve, even possibly ones that have been maliciously generated.

**HONEST SENDER AND CORRUPT RECEIVER.** This also works as in the other proof, since by simulating the honest sender we know  $t$  such that  $E = t \star E_0$ , which otherwise we knew by simulating the trusted setup functionality. ■



**Figure 9.** The twist protocol without trusted setup

## References

- ABK<sup>+</sup>21. Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Algebraic adversaries in the universal compositability framework. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 311–341, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany.
- ADMP20. Navid Alapati, Luca De Feo, Hart Montgomery, and Sikhhar Patranabis. Cryptographic group actions and applications. In Shihoh Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.
- AEK<sup>+</sup>22. Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. *Cryptology ePrint Archive*, Report 2022/770, 2022. <https://eprint.iacr.org/2022/770>.
- AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.
- BBD<sup>+</sup>22. Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D. Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E. Stange, Yan Bo Ti, Christelle Vincent, José Felipe Voloch, Charlotte Weitkämper, and Lukas Zobernig. Failing to hash into supersingular isogeny graphs. *Cryptology ePrint Archive*, Report 2022/518, 2022. <https://eprint.iacr.org/2022/518>.
- BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 370–390, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

- BDD<sup>+</sup>17. Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. Cryptology ePrint Archive, Report 2017/993, 2017. <https://eprint.iacr.org/2017/993>.
- BDGM19. Pedro Branco, Jintai Ding, Manuel Goulão, and Paulo Mateus. A framework for universally composable oblivious transfer from one-round key-exchange. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *Lecture Notes in Computer Science*, pages 78–101, Oxford, UK, December 16–18, 2019. Springer, Heidelberg, Germany.
- BLN<sup>+</sup>21. Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. High-performance multi-party computation for binary circuits based on oblivious transfer. *Journal of Cryptology*, 34(3):34, July 2021.
- BM90. Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO’89*, volume 435 of *Lecture Notes in Computer Science*, pages 547–557, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.
- BMM<sup>+</sup>22. Saikrishna Badrinarayanan, Daniel Masny, Pratyay Mukherjee, Sikhar Patranabis, Srinivasan Raghuraman, and Pratik Sarkar. Round-optimal oblivious transfer and MPC from computational CSIDH. Cryptology ePrint Archive, Report 2022/1511, 2022. <https://eprint.iacr.org/2022/1511>.
- BOB18. Paulo Barreto, Glaucio Oliveira, and Waldyr Benits. Supersingular isogeny oblivious transfer. Cryptology ePrint Archive, Report 2018/459, 2018. <https://eprint.iacr.org/2018/459>.
- Bro09. Reinier Brooker. Constructing supersingular elliptic curves. *Frontiers of Combinatorics and Number Theory*, 01 2009.
- Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.
- CD22. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975, 2022. <https://eprint.iacr.org/2022/975>.
- CDPW07. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.
- CJS14. Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.
- CLM<sup>+</sup>18. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.
- CO15. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Lecture Notes in Computer Science*, pages 40–58, Guadalajara, Mexico, August 23–26, 2015. Springer, Heidelberg, Germany.
- Cou06. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. <https://eprint.iacr.org/2006/291>.
- CPV20. Wouter Castryck, Lorenz Panny, and Frederik Vercauteren. Rational isogenies from irrational endomorphisms. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 523–548, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
- Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.
- DCW13. Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 789–800, Berlin, Germany, November 4–8, 2013. ACM Press.
- DDN14. Bernardo David, Rafael Dowsley, and Anderson C. A. Nascimento. Universally composable oblivious transfer based on a variant of LPN. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 14: 13th International Conference on Cryptology and Network Security*, volume 8813

- of *Lecture Notes in Computer Science*, pages 143–158, Heraklion, Crete, Greece, October 22–24, 2014. Springer, Heidelberg, Germany.
- DGH<sup>+</sup>20. Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 768–797, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
- DGI<sup>+</sup>19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.
- DHK<sup>+</sup>23. Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. Cryptology ePrint Archive, Report 2023/186, 2023. <https://eprint.iacr.org/2023/186>.
- dOPS20. Cyprien de Saint Guilhem, Emmanuela Orsini, Christophe Petit, and Nigel P. Smart. Semi-commutative masking: A framework for isogeny-based protocols, with an application to fully secure two-round isogeny-based OT. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20: 19th International Conference on Cryptology and Network Security*, volume 12579 of *Lecture Notes in Computer Science*, pages 235–258, Vienna, Austria, December 14–16, 2020. Springer, Heidelberg, Germany.
- dQKL<sup>+</sup>21. Victoria de Quehen, Péter Kutas, Chris Leonardi, Chloe Martindale, Lorenz Panny, Christophe Petit, and Katherine E. Stange. Improved torsion-point attacks on SIDH variants. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 432–470, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.
- DvMN08. Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. Oblivious transfer based on the McEliece assumptions. In Reihaneh Safavi-Naini, editor, *ICITS 08: 3rd International Conference on Information Theoretic Security*, volume 5155 of *Lecture Notes in Computer Science*, pages 107–117, Calgary, Canada, August 10–13, 2008. Springer, Heidelberg, Germany.
- EGL82. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO’82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.
- Fel19. Joël Felderhoff. Hard Homogenous Spaces and Commutative Supersingular Isogeny based Diffie-Hellman. Internship report, LIX, Ecole polytechnique ; ENS de Lyon, August 2019.
- Feo17. Luca De Feo. Mathematics of isogeny based cryptography. *CoRR*, abs/1711.04062, 2017.
- FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.
- GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.
- GPSV18. Steven Galbraith, Lorenz Panny, Benjamin Smith, and Frederik Vercauteren. Quantum equivalence of the DLP and CDHP for group actions. Cryptology ePrint Archive, Report 2018/1199, 2018. <https://eprint.iacr.org/2018/1199>.
- HK12. Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012.
- JAC<sup>+</sup>20. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2020. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>.
- JD11. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 19–34, Tapei, Taiwan, November 29 – December 2 2011. Springer, Heidelberg, Germany.
- Kan97. Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1997:122 – 93, 1997.
- KOS16. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 830–842, Vienna, Austria, October 24–28, 2016. ACM Press.

- KZZ22. Jonathan Katz, Cong Zhang, and Hong-Sheng Zhou. An analysis of the algebraic group model. Cryptology ePrint Archive, Report 2022/210, 2022. <https://eprint.iacr.org/2022/210>.
- LGd21. Yi-Fu Lai, Steven D. Galbraith, and Cyprien de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 213–241, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.
- Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany.
- MM22. Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Report 2022/1026, 2022. <https://eprint.iacr.org/2022/1026>.
- MMP22. Marzio Mula, Nadir Murru, and Federico Pintore. Random sampling of supersingular elliptic curves. Cryptology ePrint Archive, Report 2022/528, 2022. <https://eprint.iacr.org/2022/528>.
- MS20. Daniele Micciancio and Jessica Sorrell. Simpler statistically sender private oblivious transfer from ideals of cyclotomic integers. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 381–407, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.
- MZ22. Hart Montgomery and Mark Zhandry. Full quantum equivalence of group action DLog and CDH, and more. Cryptology ePrint Archive, Report 2022/1135, 2022. <https://eprint.iacr.org/2022/1135>.
- NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Koseraju, editor, *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Washington, DC, USA, January 7–9, 2001. ACM-SIAM.
- Pet17. Christophe Petit. Faster algorithms for isogeny problems using torsion point images. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 330–353, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.
- PSZ14. Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 797–812, San Diego, CA, USA, August 20–22, 2014. USENIX Association.
- PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.
- Rab05. Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. <https://eprint.iacr.org/2005/187>.
- Rob22. Damien Robert. Breaking SIDH in polynomial time. Cryptology ePrint Archive, Report 2022/1038, 2022. <https://eprint.iacr.org/2022/1038>.
- RS06. Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. <https://eprint.iacr.org/2006/145>.
- Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT’97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.
- Sil09. J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer New York, 2009.
- Vél71. Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus de l’Académie des Sciences de Paris*, 273:238–241, July 1971.
- Vit19. Vanessa Vitse. Simple oblivious transfer protocols compatible with supersingular isogenies. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19: 11th International Conference on Cryptology in Africa*, volume 11627 of *Lecture Notes in Computer Science*, pages 56–78, Rabat, Morocco, July 9–11, 2019. Springer, Heidelberg, Germany.
- Wes22a. Benjamin Wesolowski. Orientations and the supersingular endomorphism ring problem. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 345–371, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.
- Wes22b. Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1100–1111, 2022.



- Zha22. Mark Zhandry. To label, or not to label (in generic groups). Cryptology ePrint Archive, Report 2022/226, 2022. <https://eprint.iacr.org/2022/226>.
- ZLWR13. Bingsheng Zhang, Helger Lipmaa, Cong Wang, and Kui Ren. Practical fully simulatable oblivious transfer with sublinear communication. In Ahmad-Reza Sadeghi, editor, *FC 2013: 17th International Conference on Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 78–95, Okinawa, Japan, April 1–5, 2013. Springer, Heidelberg, Germany.

## A The Algebraic Adversary Model

Here, we briefly recall the algebraic group model by Fuchsbauer, Kiltz and Loss [FKL18], and in particular its instantiation in the UC framework, resulting in the UC-AGM as described by Abdalla et al. [ABK<sup>+</sup>21]. We also show how the AGM enables a proof of the “simplest OT” by Chou and Orlandi [CO15].

*The UC-AGM Framework.* The setting involves a group  $\mathbb{G}$  of prime order  $p$ , with known generator  $g$ . We collect those parameters in  $\mathcal{G} = (\mathbb{G}, g, p)$ . Roughly, an *algebraic adversary*, compared to a standard adversary, has an additional auxiliary tape on which it writes the representation of any group element it outputs on some other tapes. More formally, we have the following definition.

**Definition 15.** *Suppose a protocol  $\pi$  uses the group  $\mathcal{G}$  as above. A pair of environment  $\mathcal{Z}$  and adversary  $\mathcal{A}$  is said  $(\mathcal{G}, \pi)$ -algebraic if it satisfies the following conditions.*

1.  $\mathcal{A}$  has a special output tape called **algebraic tape**;
2. Whenever  $\mathcal{A}$  sends a **(backdoor,  $m$ )** message to a party and  $m$  contains an element  $h \in \mathbb{G}$ , then either
  - (a) It must provide (to a special “algebraic tape”) an algebraic representation  $X$  of  $h$ , or
  - (b)  $\mathcal{A}$  has previously received such algebraic representation from  $\mathcal{Z}$ , where the algebraic representation of  $h$  is a list  $X = [(g_1, x_1), \dots, (g_k, x_k)]$  such that  $h = \prod_{i=1}^k g_i^{x_i}$  and  $g_i$  are group elements already seen by  $\mathcal{A}$  or  $\mathcal{Z}$  in the execution of  $\pi$ .

With this definition, it is possible to restrict standard UC-emulation to algebraic adversaries and environments.

**Definition 16.** *Suppose protocols  $\pi$  and  $\phi$  involve the same group  $\mathcal{G}$ . We say that  $\pi$   $\mathcal{G}$ -AGM emulates  $\phi$  if for any efficient adversary  $\mathcal{A}$  there is an efficient simulator  $\mathcal{S}$  such that for any efficient environment  $\mathcal{Z}$  with  $(\mathcal{Z}, \mathcal{A})$  that is  $(\mathcal{G}, \pi)$ -algebraic we have that also  $(\mathcal{Z}, \mathcal{S})$  is  $(\mathcal{G}, \phi)$ -algebraic and*

$$\text{EXEC}_{\phi, \mathcal{S}, \mathcal{Z}} \approx \text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}.$$

We can apply the definition of AGM-emulations (Definition 16) to an ideal protocol  $\text{IDEAL}_{\mathcal{F}}$  and instantiate Definition 2 accordingly.

Like in standard UC, we can use dummy adversaries also in this algebraic setting; we only need to pay attention to the algebraic representations that the environment send to the adversary, because we don’t want to forward them to the actual protocol.

**Definition 17.** *Suppose the protocol  $\pi$  involves  $\mathcal{G}$ . An adversary  $\mathcal{D}$  is  $(\mathcal{G}, \pi)$ -algebraically dummy if it only forwards messages in this way:*

- For any received message of the type **(backdoor,  $m$ )** from a party  $\text{ID}$ , sends **(backdoor,  $(\text{ID}, m)$ )** to  $\mathcal{Z}$ .
- For any **(input,  $(\text{ID}, m)$ )** from  $\mathcal{Z}$ , it sends **(input,  $m'$ )** to  $\text{ID}$ , where  $m'$  is equal to  $m$ , but without all algebraic representations  $X$  of elements  $h \in \mathbb{G}$  which are inside  $m$ .

Using this definition of dummy adversary we then have the following theorem.

**Theorem 10.** *Suppose protocols  $\pi$  and  $\phi$  involve group  $\mathcal{G}$ . Then  $\pi$   $\mathcal{G}$ -AGM emulates  $\phi$  if and only if  $\pi$   $\mathcal{G}$ -AGM emulates  $\phi$  with respect to the dummy adversary.*

Observe also that since the dummy adversary doesn’t output any algebraic representation, they must all come from the environment  $\mathcal{Z}$ .

*Composition and transitivity.* It is possible to prove the composition theorem in the UC-AGM framework stated as follows.

**Theorem 11** ([ABK<sup>+</sup>21]). *Let  $\pi$  and  $\phi$  protocols involving group  $\mathcal{G}$ , such that  $\phi$  is a sub-protocol of  $\rho^\phi$ , and  $\pi$   $\mathcal{G}$ -AGM emulates  $\phi$ . Then  $\rho^\pi$  ( $\mathcal{G}, \pi, \phi$ )-AGM emulates  $\rho^\phi$ .*

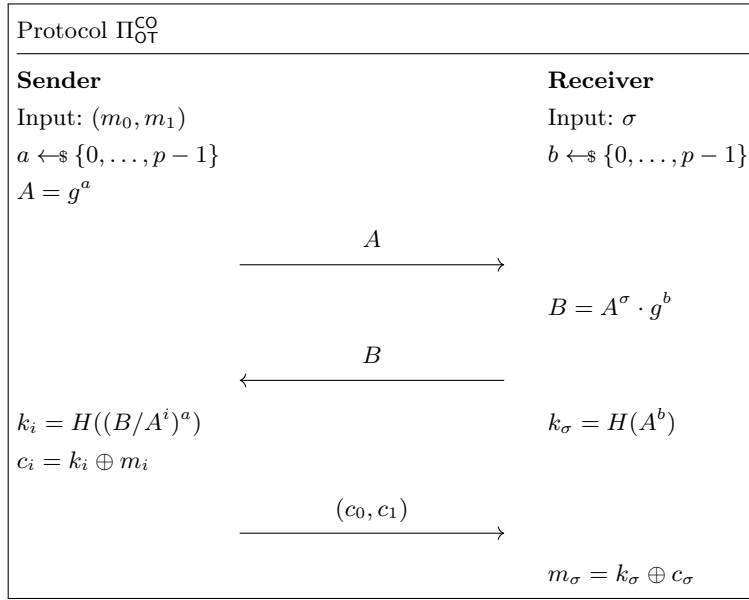
Similarly to standard UC, we have that if  $\pi, \phi'$  and  $\phi$  are protocols involving  $\mathcal{G}$ , and  $\pi$   $\mathcal{G}$ -AGM emulates  $\pi'$  (i.e.  $\pi \underset{A}{\sim} \pi'$ ) and  $\pi' \underset{A}{\sim} \phi$ , then  $\pi \underset{A}{\sim} \phi$ . However, we need to take special care when we combine the previous transitivity result with composition.

**Theorem 12** ([ABK<sup>+</sup>21]). *Suppose protocols  $\rho^{\mathcal{F}}, \pi$  and ideal functionalities  $\mathcal{F}, \mathcal{F}'$  involve the same group  $\mathcal{G}$ , such that:*

- $\text{IDEAL}_{\mathcal{F}}$  is a sub-protocol of  $\rho^{\mathcal{F}}$ ,
- $\pi$  ( $\mathcal{G}, \pi$ )-AGM realizes  $\mathcal{F}$ ,
- $\rho^{\mathcal{F}}$  ( $\mathcal{G}, \rho$ )-AGM realizes  $\mathcal{F}'$ ,

*Then, the protocol  $\rho^\pi$  AGM realizes  $\mathcal{F}'$  with respect to attackers that are both  $(\mathcal{G}, \rho)$ - and  $(\mathcal{G}, \pi)$ -algebraic.*

## A.1 The CO-OT Protocol in AGM



**Figure 10.** The Simplest OT protocol by Chou and Orlandi

Here, we will analyse the OT protocol proposed by Chou and Orlandi [CO15]. The core of the protocol is described in Figure 10, and needs a random oracle functionality, denoted here by  $H$ .

**Functionality  $\mathcal{F}_{\text{OT}}^A$**

The functionality runs with a receiver  $P_R$ , a sender  $P_S$  and an adversary  $\mathcal{S}$

- On input  $(\text{receive}, \text{sid}, \sigma)$  from  $P_R$  or  $\mathcal{S}$ , if no message with the same  $\text{sid}$  has been stored, store  $(\text{receive}, \text{sid}, \sigma)$  and notify  $\mathcal{S}$ .
- On input  $(\text{send}, \text{sid}, (m_0, m_1))$  from  $P_S$  or  $\mathcal{S}$ , if no message with the same  $\text{sid}$  has been stored, store  $(\text{send}, \text{sid}, (m_0, m_1))$  and notify  $\mathcal{S}$ .
- On input  $(\text{deliver}, \text{sid}, R)$  from the adversary, if there have been stored both messages  $(\text{receive}, \text{sid}, \sigma)$  and  $(\text{send}, \text{sid}, (m_0, m_1))$ , send  $(\text{output}, \text{sid}, m_\sigma)$  to  $P_R$ ; otherwise output  $\perp$  to  $\mathcal{S}$ .
- On input  $(\text{deliver}, \text{sid}, S)$  from the adversary, if it was previously output  $(\text{output}, \text{sid}, m_\sigma)$ , then send  $(\text{output}, \text{sid})$  to  $P_S$ ; otherwise output  $\perp$  to  $\mathcal{S}$ .

**Figure 11.** OT functionality in the AGM-UC framework

**Proof of UC-AGM security of Chou and Orlandi** The Algebraic Group Model is a key tool for proving UC security for the “simplest OT” protocol. Roughly, it uses the algebraic behaviour of the adversary both for explaining the parties’ state after adaptive corruptions and for extracting the input bit of a corrupt receiver. We will only give a sketch of the proof in the case of static corruptions. We use the protocol in Figure 10, with the functionality presented in Figure 11.

**Theorem 13.** *The protocol  $\Pi_{\text{OT}}^{\text{CO}}$  AGM-realizes the functionality  $\mathcal{F}_{\text{OT}}^A$  in the  $\mathcal{F}_{\text{RO}}$ -hybrid model under static corruptions.*

*Proof.* We construct a simulator  $\mathcal{S}$  for the dummy algebraic adversary in each of the four corruption cases. By definition, this means that all group elements output by  $\mathcal{Z}$  to  $\mathcal{S}$  must have a representation.

**CORRUPTED SENDER AND HONEST RECEIVER:** When  $\mathcal{S}$  receives  $A$  from the adversary, it also learns the value  $a$  such that  $A = g^a$ . The simulator then chooses a random  $b$ , computes  $B = g^b$ , and sends it back to  $\mathcal{A}$ .

Then it computes the key  $k = H(A^b)$ . When the adversary sends any  $(c_0, c_1)$ , the simulator sends  $(c_0 \oplus k, c_1 \oplus k)$  to the trusted party and makes it deliver to the honest receiver.

This simulates correctly since the output of the receiver is identical in both the ideal and the real execution; moreover the distributions  $g^{a\sigma+b}$  and  $g^b$  are identical, so the environment cannot distinguish between the case  $\sigma = 0$  and  $\sigma = 1$ .

**HONEST SENDER AND CORRUPTED RECEIVER:** The simulator samples  $a$ , computes  $A = g^a$  and sends it to the adversary, which responds with an arbitrary element  $B$ ; since  $\mathcal{A}$  is algebraic, it must also output a representation  $B = A^x g^y$ .

If  $x \in \{0, 1\}$ , the simulator queries the functionality with this bit, and sets  $m_x$  to the retrieved value; in all other cases (i.e.  $i = 1 - x$ , or both 0 and 1 if  $x \notin \{0, 1\}$ ), it sets  $m_i$  to null. It also samples random  $c_0, c_1$ .

The simulator also runs the random oracle, and checks the queries made to it. In particular, upon learning  $B$ , it retroactively checks all queries for the values  $B^a g^{-ia^2}$ : if  $m_i$  is null the simulator aborts, otherwise it sets  $c_i = k_i \oplus m_i$ , where  $k_i$  was the answer of the query; it also does this for future queries, this time by computing the answer as  $k_i = c_i \oplus m_i$ . This means that  $\mathcal{S}$  aborts precisely when  $x \in \{0, 1\}$  and  $\mathcal{A}$  queries for both  $B^a$  and  $B^a g^{-a^2}$ , or if  $x \notin \{0, 1\}$  and  $\mathcal{A}$  queries at least one of  $B^a$  or  $B^a g^{-a^2}$ .

The simulator concludes the simulation by sending  $(c_0, c_1)$  to the adversary.

Notice that when  $\mathcal{S}$  does not abort, the simulation is perfect. Thus, the proof follows from this claim:

*Claim.*  $\mathcal{S}$  aborts with negligible probability if the discrete logarithm is hard.

*Proof.* Suppose we want to solve the discrete logarithm problem  $A = g^a$ , using  $\mathcal{A}$  as an oracle. We feed  $\mathcal{A}$  the element  $A$  as coming from the simulator. Notice that now the simulator cannot check what is the query that makes it abort, so the solver for the discrete logarithm problem analyzes all queries made by  $\mathcal{A}$ , and for each of them tries to solve the equation in  $z$  and checks if  $g^z = A$ , thus finding the secret exponent.

- Case  $x \notin \{0, 1\}$ . Suppose  $\mathcal{A}$  queries  $B^z$ . Being algebraic, it must know a representation  $B^z = A^s g^t$ . But this means that  $g^{z^2 x + zy} = (A^x g^y)^z = B^z = g^{sz+t}$ , i.e.

$$z^2 x + z(y - s) - t \equiv 0 \pmod{p}$$

from which we can compute  $z$ . In the other case we have that  $B^z g^{-z^2} = A^s g^t$ , for which the equation is  $z^2(x - 1) + zy \equiv sz + t \pmod{p}$ , which also has a solution.

- Case  $x = 0$ . The adversary  $\mathcal{A}$  has queried  $B^z g^{-z^2}$ , for which it knows a representation  $A^s g^t$ . Then it gets the equation  $zy - z^2 \equiv sz + t \pmod{p}$ , which has a solution.
- Case  $x = 1$ . The adversary  $\mathcal{A}$  has queried  $B^z$ , and represents it as  $A^s g^t$ . Then the equation is  $z^2 + zy \equiv sz + t \pmod{p}$ , which also has a solution.

This concludes the proof of the claim.

HONEST SENDER AND HONEST RECEIVER: This simulation can be constructed putting together both simulations above, as we did in the proof of the toy protocol.

This concludes the proof since if the simulator doesn't abort, the simulation is perfect, given that the keys queried from the random oracle statistically hide the messages. Finally we observe that all the simulators we have constructed are algebraic themselves.