Round-Optimal Oblivious Transfer and MPC from Computational CSIDH

Saikrishna Badrinarayanan[*] Snap

Daniel Masny[†] Meta

Pratyay Mukherjee[‡] Swirlds Labs

Sikhar Patranabis[§] IBM Research India

Srinivasan Raghuraman Visa Research

Pratik Sarkar[¶] Boston University

Abstract

We present the first round-optimal and plausibly quantum-safe oblivious transfer (OT) and multi-party computation (MPC) protocols from the computational CSIDH assumption – the weakest and most widely studied assumption in the CSIDH family of isogeny-based assumptions. We obtain the following results:

- The *first* round-optimal maliciously secure OT and MPC protocols in the *plain model* that achieve (black-box) simulation-based security while relying on the computational CSIDH assumption.
- The *first* round-optimal maliciously secure OT and MPC protocols that achieves Universal Composability (UC) security in the presence of a trusted setup (common reference string plus random oracle) while relying on the computational CSIDH assumption.

Prior plausibly quantum-safe isogeny-based OT protocols (with/without setup assumptions) are either not round-optimal, or rely on potentially stronger assumptions.

We also build a 3-round maliciously-secure OT extension protocol where each base OT protocol requires only 4 isogeny computations. In comparison, the most efficient isogeny-based OT extension protocol till date due to Lai et al. [Eurocrypt 2021] requires 12 isogeny computations and 4 rounds of communication, while relying on the same assumption as our construction, namely the reciprocal CSIDH assumption.

[*]Work done while the author was affiliated with Visa Research.

[†]Work done while the author was affiliated with Visa Research.

[‡]Work done while the author was affiliated with Visa Research.

[§]Most of the work was done while the author was affiliated with Visa Research.

[¶]Supported by NSF Awards 1931714, 1414119, and the DARPA SIEVE program. Part of the work was done while the author was an intern at VISA Research.

Contents

1 Introduction

Oblivious transfer (OT) [Rab05, EGL82] is an interactive protocol between two parties: a sender and a receiver. Informally speaking, an OT protocol involves a *sender* holding two messages m_0 and m_1, and a receiver holding a bit $b \in \{0, 1\}$. At the end of the protocol, the receiver should only learn the message m_b and nothing about the other message m_{1-b}, while the sender should learn nothing about the bit b. OT serves as a fundamental building block in cryptography [Kil88], particularly in secure multi-party computation (MPC) [Yao86, IKO+11, BL18, GS18]. Round optimal OT protocols imply round-optimal MPC protocols [BL18, GS18, CCG+20] and hence are always desirable.

Quantum-Safe OT. With steady progress in quantum computing, the study of post-quantum cryptography has gained significant momentum in recent years, especially in light of Shor's algorithm [Sho94], which breaks traditional cryptographic assumptions such as factoring and discrete log. OT protocols are known from various plausibly quantum-safe assumptions such as lattices [PVW08, BD18, MR19], codes [DvMN08, DNM12, MR19], and isogenies of elliptic curves [BOB18, Vit18, LGdSG21]. Unfortunately, many isogeny-based OT constructions [BOB18, dSGOPS20, Vit18] are now (classically) broken in light of the recent attacks on the Supersingular Isogeny Diffie-Hellman (SIDH) assumption [CD22, MM22]. Hence, the only plausibly quantum-safe isogeny-based OT constructions are the ones based on the Commutative SIDH (CSIDH) [CLM+18] family of isogeny-based assumptions, which are not affected by the recent attacks on SIDH.

The CSIDH Family of Assumptions. The CSIDH family of (plausibly quantum-safe) isogeny-based assumptions includes the computational CSIDH assumption [CLM+18] (the CSIDH-equivalent of the traditional CDH assumption), the decisional CSIDH assumption [CSV20, ADMP20, BKW20] (the CSIDH-equivalent of the traditional DDH assumption), the reciprocal CSIDH assumption [LGdSG21], and certain variants of these assumptions [AEK+22]. Of these, the computational CSIDH assumption is the weakest assumption (equivalently, the hardest problem to solve). The decisional CSIDH assumption implies the computational CSIDH assumption, and has been shown to be broken for certain families of elliptic curves [CSV20]. Finally, the reciprocal CSIDH assumption is only *quantum-equivalent* to the computational CSIDH assumption; the corresponding classical equivalence is not known (see discussion in [LGdSG21]).

OT from CSIDH-based Assumptions. Many recent works have constructed OT protocols from the CSIDH family of isogeny-based assumptions. We broadly categorize these OT constructions as: (i) OT protocols in the *plain model*, i.e., without any (trusted) setup assumptions, or (ii) OT protocols in the *setup model*, i.e., assuming the existence of some (trusted) setup and/or random oracles.

In the plain model, there exist round-optimal OT protocols achieving various security notions from the decisional CSIDH assumption [ADMP20, KM20] and the reciprocal CSIDH assumption [BPS22]. We present a summary of these protocols in Table 1. In the setup model, round-optimal OT protocols are known from the decisional CSIDH assumption [ADMP20, BKW20, AMPS21]. A recent work by Lai et al. [LGdSG21] proposed an elegant OT protocol from the reciprocal CSIDH assumption; however, their construction is *not* round-optimal. We summarize these protocols in Table 2.

Notably, there exist no (round-optimal) OT protocols in the plain/setup model from the computational CSIDH assumption, which is the weakest (and most widely studied) assumption in the CSIDH family of isogeny-based assumptions. This motivates us to ask the following question:

Table 1: Comparison of plausibly quantum-safe maliciously secure OT protocols in the plain model from the CSIDH family of isogeny-based assumptions

Protocol	Computational Assumption	Rounds	Security Model
[ADMP20]-1	decisional CSIDH	2	semantic
[BPS22]-1	reciprocal CSIDH	3	semantic
[KM20]	decisional CSIDH	4	simulation-secure
[BPS22]-2	reciprocal CSIDH	4	simulation-secure
Our Protocol-1	**computational CSIDH**	**4**	**simulation-secure**

Table 2: Comparison of plausibly quantum-safe maliciously secure OT protocols in the setup model from the CSIDH family of isogeny-based assumptions. The protocols of [ADMP20, AMPS21] are in the CRS model. All other protocols are in the CRS+random oracle model.

Protocols	Computational Assumption	Rounds	Security Model
[ADMP20]-2	decisional CSIDH	2	UC-secure
[BKW20]	decisional CSIDH	2	UC-secure
[AMPS21]	decisional CSIDH	2	UC-secure
[LGdSG21]-1	reciprocal CSIDH	3	simulation-secure
[LGdSG21]-2	reciprocal CSIDH	4	UC-secure
Our Protocol-2	**computational CSIDH**	**2**	**UC-secure**

Can we design round-optimal OT protocols from computational CSIDH?

1.1 Our Contributions

In this paper, we answer the above question in the affirmative by presenting the first round-optimal, maliciously secure, and plausibly quantum safe OT protocols in various settings from the computational CSIDH assumption. In particular, we propose two new round-optimal maliciously secure OT protocols in the plain and common reference string[1] (CRS) models, while relying on the computational CSIDH assumption. These also yield the first round-optimal MPC protocols in the respective settings from the computational CSIDH assumption. Our main contributions can be summarized as follows.

Round Optimal OT and MPC in the Plain Model. We propose the *first* round-optimal (4-round) OT protocol in the plain model while relying on the computational CSIDH assumption. Our construction satisfies perfect correctness and simulation-based security against malicious corruption of parties, which is the strongest notion of OT security that is achievable in the plain model. Our result is captured by the following (informal) theorem.

Theorem 1. (Informal) *Assuming computational CSIDH, there exists a 4-round OT protocol in the plain model that achieves perfect correctness and (black-box) simulation-security against malicious corruption of parties.*

In Table 1, we present a comparison of our proposed OT construction with known constructions of round-optimal OT in the plain model from the CSIDH family of assumptions. Additionally, by invoking known relationships between round-optimal OT and MPC in the plain model from [CCG$^+$20], we achieve the following (informal) corollary.

[1]The setup string is structured and it is sampled from a given distribution.

Corollary 1. (Informal) *Assuming computational CSIDH, there exists a 4-round MPC protocol in the plain model with (black-box) simulation-security against malicious corruption of parties.*

This is the first round optimal MPC protocol achieving (black-box) simulation security in the plain model from the computational CSIDH assumption.

Round-Optimal OT and MPC assuming Trusted Setup. We propose the *first* round-optimal (2-round) OT protocol in the CRS plus random oracle model[2] while relying on the computational CSIDH assumption. Our construction satisfies perfect correctness and universal composability (UC)-security against malicious corruption of parties, which is the strongest notion of OT security that is achievable in the trusted setup model. Informally, we prove the following theorem.

Theorem 2. (Informal) *Assuming that the computational CSIDH assumption holds, there exists a 2-round OT protocol in the CRS plus random oracle model that is UC-secure against malicious corruption of parties.*

In Table 2, we present a comparison of our proposed OT construction with known constructions of round-optimal OT in the trusted setup model from the CSIDH family of assumptions. Finally, by invoking known relationships between round-optimal OT and MPC from [GS18], we achieve the following (informal) corollary.

Corollary 2. (Informal) *Assuming that the computational CSIDH assumption holds, there exists a 2-round MPC protocol in the CRS plus random oracle model that is UC-secure against malicious corruption of parties.*

This yields the *first* construction of round-optimal MPC in the CRS plus random oracle model from the computational CSIDH assumption.

Efficient OT Extension. As an additional contribution, we propose the first UC-secure OT extension protocol that relies on the computational CSIDH assumption. Concretely, we show that an optimized variant of the recent 4-round OT protocol due to Lai et al. [LGdSG21] can be plugged into the OT extension compiler due to Canetti et al. [CSW20] to build a UC-secure 3-round *OT extension protocol* in the random oracle model. This yields the most efficient (to our knowledge) UC-secure OT extension protocol currently known from isogeny-based assumptions.[3]

Our construction of OT extension builds upon a maliciously secure base OT protocol that requires a total of 4 isogeny computations. On the other hand, the state-of-the-art 4-round maliciously secure protocol of [LGdSG21] incurs 12 isogeny computations, while relying on the same hardness assumption as our construction (the reciprocal CSIDH assumption).

1.2 Related Work

In this section, we review existing OT constructions from other plausibly post-quantum secure assumptions, such as lattices and codes, as well as constructions of OT relying on generic cryptographic primitives that can be instantiated from these assumptions. We then elaborate on the prior constructions of isogeny-based OT from the CSIDH family of assumptions. See Tables 1 and 2 for a comparison our results with these prior OT protocols in the plain and setup models, respectively.

[2]The random oracles in our protocol are local to each session.

[3]We note that while prior works on OT from isogenies do not explicitly construct OT extension protocols, they do yield base OT protocols that can be converted in a generic manner into full-fledged OT extension protocols.

Lattice-based OT. To the best of our knowledge, the first lattice-based oblivious transfer protocol was designed by Peikert, Vaikuntanathan and Waters [PVW08], that relies on LWE [Reg05]. Their OT protocol follows a more generic framework on dual encryption and achieves round-optimality as well as UC security in the CRS model. A recent result of Quach [Qua20] improves the [PVW08] construction so that the CRS can be reused by multiple OT executions. Another recent work by Büscher et al. [BDK$^+$20] provided an instantiation of a lattice-based OT from additive homomorphic encryption. The work of [AMPS21] presents the first adaptively secure OT protocol from LPN in the crs model. Their protocol is UC-secure and requires two rounds. The OT construction of Brakerski and Döttling [BD18] provided the first two-round SSP OT (without a CRS).

An alternative to constructing an OT is to construct an oblivious pseudorandom function which implies [JL09] an OT. Albrecht, Davidson, Deo and Smart [ADDS21] showed how to construct an oblivious pseudorandom function from ideal lattices using non-interactive zero-knowledge arguements which can be obtained efficiently in the random oracle model.

Code-based OT. There are two OT constructions based on code-based assumptions [DvMN08, DNM12]. Both of these constructions use the specific assumption underlying the McEliece cryptosystems [McE78]. Among these, only the latter achieves UC security. Recently, Bitansky and Freizeit [BF22] showed how to realize a statistically sender-private (SSP) OT protocol with semantic security against a computationally bounded sender and an unbounded receiver while relying on the learning with parity (LPN) assumption plus Nissan Wigderson style derandomization.

Generic OT constructions. Generic approaches to realize [BGJ$^+$18, MR19, FMV19, DGH$^+$20] OT rely on public-key encryption schemes with specific properties. Unfortunately, known public-key encryption schemes from isogeny-based assumptions (including the CSIDH family of assumptions) do not satisfy any of these properties. For example, to use any isogeny-based PKE in the framework of [MR19], one inherently needs the ability to hash into a curve in the family of supersingular elliptic curves, which is not known so far (see [Pet17, DMPS19, CPV20] for more details). For the constructions of Badrinarayanan et al. [BGJ$^+$18] and Friolo et al. [FMV19] in the plain model, one needs a PKE with dense public-key space – this is again not known to exist from isogeny-based assumptions. Döttling et al. [DGH$^+$20] provided a generic approach to obtain 2-round UC-secure OT in the CRS model from protocols satisfying very mild form of security, known as elementary OT – this gives 2-round OT from LPN [ACPS09].

Prior Isogeny-based OT. Prior works [BOB18, dSGOPS20, Vit18, BKW20] have realized isogeny-based OT constructions from the well-known SIDH assumption and its variants. Unfortunately, these constructions are now (classically) broken in light of the recent attacks on the SIDH assumption [CD22, MM22]. The construction of [BKW20] was, in fact, broken in its original form by an earlier attack proposed in [BKM$^+$21].

Prior works have realized OT protocols in the plain model achieving various security notions from the decisional CSIDH assumption [ADMP20, KM20] and the reciprocal CSIDH assumption [BPS22]. The authors of [ADMP20] showed how to construct a 2-round SSP OT protocol with semantic security against a computationally bounded sender and an unbounded receiver from the decisional CSIDH assumption. The authors of [KM20] showed how to construct a 4-round OT protocol with full-fledged simulation security from any 2-round SSP OT protocol. The authors of [BPS22] showed how to construct a 3-round statistically receiver-private (SRP) OT protocol with semantic security against a computationally bounded receiver and an unbounded sender from the reciprocal CSIDH assumption. They also showed a construction of 4-round OT protocol with full-fledged simulation security from any 3-round SRP OT protocol. See Table 1 for a comparison of our proposed OT protocol in the plain model with these prior OT protocols.

In the setup model, round-optimal OT protocols are known from the decisional CSIDH assumption [ADMP20, BKW20, AMPS21]. The OT construction of [BKW20] was not explicitly described, but follows implicitly from the construction of oblivious PRF from decisional CSIDH (plus random oracles) in the same paper. The work of [AMPS21] presents the first adaptively secure OT protocol from isogenies. Their protocol is round optimal and relies on decisional CSIDH assumption. The recent work by Lai et al. [LGdSG21] proposed an elegant OT protocol from the reciprocal CSIDH assumption (plus random oracles); however, the simulation-secure and UC-secure versions of their construction require 3 rounds and 4 rounds, respectively, and are hence not round-optimal.

2 Preliminaries

Notation. For $a \in \mathbb{N}$ such that $a \geq 1$, we denote by $[a]$ the set of integers lying between 1 and a (both inclusive). We use κ to denote the security parameter, and denote by $\mathsf{poly}(\kappa)$ and $\mathsf{negl}(\kappa)$ any generic (unspecified) polynomial function and negligible function in κ, respectively. For a finite set S, we use $s \leftarrow_R S$ to sample uniformly from the set S. For a probability distribution $\mathcal{D}$ on a finite set S, we use $s \leftarrow_R \mathcal{D}$ to sample from $\mathcal{D}$. We use the notations $\overset{s}{\approx}$ and $\overset{c}{\approx}$ to denote statistical and computational indistinguishability of distributions, respectively.

2.1 Basic Cryptographic Primitives

Weak Unpredictable Function (wUF). Let K, X, and Y be sets indexed by κ. A weak unpredictable function (wUF) family is a family of efficiently computable functions $\{F(k,\cdot) : X \to Y\}_{k \in K}$ such that for all PPT adversaries $\mathcal{A}$ we have the following:

$$\Pr[\mathcal{A}^{F_k^\$}(1^\kappa, x^*) = F(k, x^*)] \leq \mathsf{negl}(\kappa),$$

where $k \leftarrow_R K$, $x^* \leftarrow_R X$, and $F_k^\$$ is a *randomized* oracle that when queried samples $x \leftarrow_R X$ and outputs $(x, F(k,x))$.

Weak Pseudorandom Function (wPRF). Let K, X, and Y be sets indexed by κ. A weak pseudorandom function (wPRF) is a family of efficiently computable functions $\{F(k,\cdot) : X \to Y\}_{k \in K}$ such that for all PPT adversaries $\mathcal{A}$ we have the following:

$$\left|\Pr[\mathcal{A}^{F_k^\$}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\pi^\$}(1^\kappa) = 1]\right| \leq \mathsf{negl}(\kappa),$$

where $k \leftarrow_R k$, $F_k^\$$ is a randomized oracle that when queried samples $x \leftarrow_R X$ and outputs $(x, F(k,x))$, and $\pi^\$$ is a randomized oracle that when queried samples $x \leftarrow_R X$ and $y \leftarrow_R Y$, and outputs (x, y).

2.2 Cryptographic Group Actions

In this section we recall the definitions of cryptographic group actions from [ADMP20]. We note here that the authors of [ADMP20] use the definitions of Brassard and Yung [BY91] and Couveignes [Cou06] as starting points to provide definitions that allow for easy use of isogenies (in particular, isogeny families such as CSIDH [CLM+18] and CSI-FiSh [BKV19]) in cryptographic protocols. We begin by recalling the definition of a group action.

Definition 1. (Group Action [BY91, Cou06, ADMP20]). *A group G is said to* act on *a set X if there is a map $\star : G \times X \to X$ that satisfies:*

1. *Identity: If e is the identity element of G, then for any $x \in X$, we have $e \star x = x$.*
2. *Compatibility: For any $g, h \in G$ and any $x \in X$, we have $(gh) \star x = g \star (h \star x)$.*

Throughout this paper, we use the abbreviated notation $(G, X, \star)$ to denote a group action.

Remark 1. *If $(G, X, \star)$ is a group action, for any $g \in G$ the map $\pi_g : x \mapsto g \star x$ defines a permutation of X.*

Properties of Group Actions. We consider group actions $(G, X, \star)$ that satisfy one or more of the following properties:

1. *Abelian:* The group G is abelian.
2. *Transitive:* For every $x_1, x_2 \in X$, there exists a group element $g \in G$ such that $x_2 = g \star x_1$. For such a transitive group action, the set X is called a *homogeneous space* for G.
3. *Faithful:* For each group element $g \in G$, either g is the identity element or there exists a set element $x \in X$ such that $x \neq g \star x$.
4. *Free:* For each group element $g \in G$, g is the identity element if and only if there exists some set element $x \in X$ such that $x = g \star x$.
5. *Regular: Both* free *and* transitive.

Remark 2. *If a group action is regular, then for any $x \in X$, the map $f_x : g \mapsto g \star x$ defines a bijection between G and X; in particular, if G (or X) is finite, then we must have $|G| = |X|$.*

Effective Group Action (EGA). We now recall the definition of an *effective* group action (abbreviated throughout as an EGA) from [ADMP20]. At a high level, an EGA is an abelian and regular group action with certain special computational properties that allow it to be useful for cryptographic applications. Formally, an abelian and regular group action $(G, X, \star)$ is *effective* if the following properties are satisfied:

1. The group G is finite and there exist efficient (PPT) algorithms for:
   (a) Membership testing, i.e., to decide if a given bit string represents a valid group element in G.
   (b) Equality testing, i.e., to decide if two bit strings represent the same group element in G.
   (c) Sampling, i.e., to sample an element g from a distribution G on G. In this paper, We consider distributions that are statistically close to uniform.
   (d) Operation, i.e., to compute gh for any $g, h \in G$.
   (e) Inversion, i.e., to compute g^{-1} for any $g \in G$.
2. The set X is finite and there exist efficient algorithms for:
   (a) Membership testing, i.e., to decide if a bit string represents a valid set element.

(b) Unique representation, i.e., given any arbitrary set element $x \in X$, compute a string $\hat{x}$ that canonically represents x.

3. There exists a distinguished element $x_0 \in X$, called the *origin*, such that its bit-string representation is known.

4. There exists an efficient algorithm that given (some bit-string representations of) any $g \in G$ and any $x \in X$, outputs $g \star x$.

Restricted Effective Group Action (REGA). From the point of view of cryptographic applications, one can view EGA as an abstraction that captures the CSI-FiSh [BKV19] family of isogenies, where we can compute the group action operation $\star$ efficiently for any element g in the group G. However, this is not the case for the CSIDH family of isogenies [CLM+18], where we can only compute the group action operation $\star$ efficiently for "certain" elements in the group G (more specifically, a generating set of small cardinality). To model such families of isogenies, the authors of [ADMP20] introduced a weaker or *restricted* variant of EGA (abbreviated throughout as REGA).

Let $(G, X, \star)$ be an abelian and regular group action where G and X are both finite, and let $\mathbf{g} = (g_1, \ldots, g_n)$ be a (not necessarily minimal) generating set for the group G, where $n = \mathsf{poly}(\log(|G|))$. The action is said to be $\mathbf{g}$-*restricted effective*, if the following properties are satisfied:

- There exist efficient algorithms for:
  1. Membership testing, i.e., to decide if a bit string represents a valid set element.
  2. Unique representation, i.e., to compute a string $\hat{x}$ that canonically represents any given set element $x \in X$.
- There exists a distinguished element $x_0 \in X$, called the *origin*, such that its bit-string representation is known.
- There exists an efficient algorithm that given any $i \in [n]$ and any bit string representation of $x \in X$, outputs $g_i \star x$ and $g_i^{-1} \star x$.

When the group G is abelian, we represent a *word* on $\mathbf{g}$ as a vector in $\mathbf{a} \in \mathbb{Z}^n$, canonically mapped to G by

$$\mathbf{a} = (a_1, \ldots, a_n) \mapsto \prod_{i=1}^{n} g_i^{a_i}.$$

EGA vs REGA. Note that unlike EGA, an REGA does not allow efficiently sampling directly from the group G. In addition, an REGA is limited to evaluations of the form $g_i \star x$. A natural question is to ask is whether this limits the usefulness of REGA in cryptographic protocols, in compared to the more "nicely behaved" EGA. In other words, given a cryptographic protocol built from an EGA that requires computing actions using uniformly sampled group elements, can we have a counterpart protocol built from an REGA that remains computationally efficient (at least in an asymptotic sense)? It turns out that this is indeed the case provided that the protocol remains secure as long as the action is computed using a group element that is sampled from a distribution *statistically close to uniform*.

More concretely, given an REGA, one can still use the generating set $\mathbf{g}$ to sample a group element g from a distribution that is *statistically* close to uniform over G, while retaining the ability to compute $g \star x$ efficiently for any $x \in X$. The idea is to represent g as a word on $\mathbf{g}$ using a vector $\mathbf{a} \in \mathbb{Z}^n$, as described above, where $\mathbf{a}$ is sampled from a discrete Gaussian distribution [DG19]. In particular, the authors of [DG19] argue that such a representation plausibly yields a group element g that is distributed statistically close to uniform.

Throughout this paper, we focus on protocols based on group actions where it is sufficient to rely on action computations where the group element g is sampled from a distribution that is *statistically* close to uniform over G. Hence, all of our protocols can be instantiated using both EGA and REGA (and hence from both CSI-FiSh [BKV19] and CSIDH [CLM+18]). For simplicity of representation, we describe our constructions from an EGA; the corresponding REGA-based constructions follow analogously.

Hardness Assumptions over EGA. We now define certain hardness assumptions pertaining to an EGA following conventions introduced in [ADMP20].

Definition 2. (Weak Unpredictable EGA [ADMP20]). *An EGA $(G, X, \star)$ is weakly unpredictable if the family of functions (more specifically, permutations) $\{\pi_g : X \to X\}_{g \in G}$ is weakly unpredictable, where π_g is defined as $\pi_g : x \mapsto g \star x$.*

Definition 3. (Weak Pseudorandom EGA [ADMP20]). *An EGA $(G, X, \star)$ is weakly pseudorandom if the family of functions (more specifically, permutations) $\{\pi_g : X \to X\}_{g \in G}$ is weakly pseudorandom, where π_g is defined as $\pi_g : x \mapsto g \star x$.*

Throughout this paper, we will use the abbreviations wU-EGA and wPR-EGA to refer to a weak unpredictable and weak pseudorandom (abelian and regular) EGA, respectively. We can similarly define wU-REGA and wPR-REGA, where in the corresponding definitions, all group elements are sampled from a distribution that is statistically close to uniform. Finally, we state the following theorem (imported from [ADMP20]).

Theorem 3. ([ADMP20]). *Assuming that the computational (resp., decisional) CSIDH assumption holds, there exists a wU-REGA (resp., wPR-REGA).*

All of the protocols proposed in this paper can be instantiated using both EGA and REGA (and hence from both CSI-FiSh [BKV19] and CSIDH [CLM+18]). For simplicity of representation, we describe our constructions from an EGA; the corresponding REGA-based constructions follow analogously.

2.3 Oblivious Transfer (OT)

In this section, we present preliminary background material on oblivious transfer (OT) protocols.

The Ideal Functionality for OT. The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for any OT protocol is described in Figure 1. We adopt this description essentially verbatim from prior works [CLOS02, PVW08, DGH+20].

2.3.1 Two-Round Oblivious Transfer in the CRS Model

We first formally define a two-round oblivious transfer (OT) protocol in the CRS model. A two-round OT protocol in the CRS model is a tuple of four algorithms of the form $\mathsf{OT} = (\mathsf{Setup}, \mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ described below:

Figure 1: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for Oblivious Transfer

$\mathcal{F}_{\mathsf{OT}}$

$\mathcal{F}_{\mathsf{OT}}$ interacts with an ideal sender S and an ideal receiver R as follows:

– On input $(\mathsf{Choose}, \mathsf{rec}, \mathsf{sid}, b)$ from R where $b \in \{0,1\}$; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S.

– On input $(\mathsf{Transfer}, \mathsf{sen}, \mathsf{sid}, (\mathsf{m}_0, \mathsf{m}_1))$ from S with $\mathsf{m}_0, \mathsf{m}_1 \in \{0,1\}^n$, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (\mathsf{m}_0, \mathsf{m}_1))$ is recorded and a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, send $(\mathsf{sent}, \mathsf{sid}, \mathsf{m}_b)$ to R and $(\mathsf{sent}, \mathsf{sid})$ to S. Ignore future messages with the same sid.

Figure 2: The ideal functionality $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{CRS}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{CRS}}^{\mathcal{D}}$ (parameterized by a distribution $\mathcal{D}$) is run by parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$ as follows:

– Whenever receiving a message of the form (sid, P_i, P_j) from a party P_i, sample $\mathsf{crs} \leftarrow_R \mathcal{D}$ and send $(\mathsf{sid}, \mathsf{crs})$ to the party P_i and send $(\mathsf{sid}, \mathsf{crs}, P_i, P_j)$ to the adversary $\mathcal{S}$.

– Whenever receiving the message of the form (sid, P_i, P_j) from a party P_j, send $(\mathsf{sid}, \mathsf{crs})$ to both the party P_j and the adversary $\mathcal{S}$.

- $\mathsf{Setup}(1^\kappa)$: Takes as input the security parameter κ and outputs a CRS string crs and a trapdoor td.[4]

- $\mathsf{OTR}(\mathsf{crs}, b \in \{0,1\})$: Takes as input the crs and a bit $b \in \{0,1\}$, and outputs the receiver's message ot_1 and the receiver's (secret) internal state st.

- $\mathsf{OTS}(\mathsf{crs}, \mathsf{ot}_1, \mathsf{m}_0, \mathsf{m}_1)$: Takes as input the crs, the receiver's message ot_1, a pair of input strings $(\mathsf{m}_0, \mathsf{m}_1)$, and outputs the sender's message ot_2.

- $\mathsf{OTD}(\mathsf{crs}, \mathsf{st}, \mathsf{ot}_2)$: Takes as input the crs, the sender's message ot_2, and the receiver's internal state st, and outputs a message string m'.

Correctness. A two-round OT protocol in the CRS model is said to be correct if for any $b \in \{0,1\}$ and any $(\mathsf{m}_0, \mathsf{m}_1)$, letting $(\mathsf{crs}, \mathsf{td}) \leftarrow_R \mathsf{Setup}(1^\kappa)$ and $(\mathsf{ot}_1, \mathsf{st}) \leftarrow_R \mathsf{OTR}(\mathsf{crs}, b)$, we have $\mathsf{OTD}(\mathsf{crs}, \mathsf{st}, \mathsf{OTS}(\mathsf{crs}, \mathsf{ot}_1, \mathsf{m}_0, \mathsf{m}_1)) = \mathsf{m}_b$.

UC Security. For UC security of OT against malicious adversaries (in the static corruption setting), we directly use Canetti's UC security framework for static corruptions [Can01].

Following the standard notation associated with Canetti's UC security framework [Can01], we use $\mathcal{Z}$ to denote the underlying environment. For a real protocol π and an adversary $\mathcal{A}$, we use $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ to denote the real-world ensemble. Also, for an ideal functionality $\mathcal{F}$ and an adversary $\mathcal{S}$, we use $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ to denote the corresponding ideal-world ensemble.

[4]For standard two-round OT protocols, the setup algorithm need not output a trapdoor td, but we include it for certain security properties described subsequently.

The Ideal Functionality for CRS. For OT protocols in the CRS model, we also describe the ideal functionality $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{D}}$ (parameterized by a distribution $\mathcal{D}$ from which the CRS string is sampled) in Figure 2. Again, we adopt this description essentially verbatim from prior works [CR03, PVW08, DGH+20].

Receiver's UC Security. We say that an OT protocol π_{OT} satisfies receiver's UC security if for any (malicious) adversary $\mathcal{A}$ corrupting the sender, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, we have:

$$\text{EXEC}_{\pi_{\mathsf{OT}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathcal{S},\mathcal{Z}},$$

where the ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$ is as described in Figure 1.

Sender's UC Security. We say that an OT protocol π_{OT} satisfies sender's UC security if for any (malicious) adversary $\mathcal{A}$ corrupting the receiver, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, we have:

$$\text{EXEC}_{\pi_{\mathsf{OT}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathcal{S},\mathcal{Z}},$$

where the ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$ is again as described in Figure 1.

Note that in the above descriptions, we adopt the same style of security definitions as was used in prior work [CLOS02, PVW08, DGH+20]. For all of our constructions of 2-round OT protocol in the CRS model in Section 3, we prove UC-security as per the definition described above.

2.3.2 Four-Round Oblivious Transfer in the Plain Model

We also formally define a four-round oblivious transfer (OT) protocol in the plain model. A four-round OT protocol in the plain model is a tuple $\mathsf{OT} = (\mathsf{OTR}_1, \mathsf{OTS}_1, \mathsf{OTR}_2, \mathsf{OTS}_2, \mathsf{OTD})$ described below:

- $\mathsf{OTR}_1(1^\kappa, b)$: Given κ and a bit $b \in \{0,1\}$, output message ot_1 and (secret) receiver state st_R.
- $\mathsf{OTS}_1(1^\kappa, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: Given κ, a pair of strings $(\mathsf{m}_0, \mathsf{m}_1)$, and a message ot_1, output message ot_2 and (secret) sender state st_S.
- $\mathsf{OTR}_2(\mathsf{st}_\mathsf{R}, \mathsf{ot}_2)$: Given receiver state st_R and a message ot_2, output message ot_3 and an updated receiver state st_R.
- $\mathsf{OTS}_2(\mathsf{st}_\mathsf{S}, \mathsf{ot}_3)$: Given sender state st_S and message ot_3, output message ot_4.
- $\mathsf{OTD}(\mathsf{st}_\mathsf{R}, \mathsf{ot}_4)$: Given receiver state st_R and message ot_4, output string m'.

Correctness. A four-round OT protocol in the plain model is said to be correct if for any bit $b \in \{0,1\}$ and any pair of strings $\mathsf{m}_0, \mathsf{m}_1$, letting

$$\begin{aligned} (\mathsf{ot}_1, \mathsf{st}_\mathsf{R}) = \mathsf{OTR}_1(1^\kappa, b) \quad &, \quad (\mathsf{ot}_2, \mathsf{st}_\mathsf{S}) = \mathsf{OTS}_1(1^\kappa, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1), \\ (\mathsf{ot}_3, \mathsf{st}_\mathsf{R}) = \mathsf{OTR}_2(\mathsf{st}_\mathsf{R}, \mathsf{ot}_2) \quad &, \quad \mathsf{ot}_4 = \mathsf{OTS}_2(\mathsf{st}_\mathsf{S}, \mathsf{ot}_3), \end{aligned}$$

and finally

$$\mathsf{m}' = \mathsf{OTD}(\mathsf{st}_\mathsf{R}, \mathsf{ot}_4),$$

we have $\mathsf{m}' = \mathsf{m}_b$ with overwhelming probability.

Simulation Security in the Plain Model. We say that any 4-round OT protocol in the plain model is simulation-secure against maliciously corrupt parties if it implements the $\mathcal{F}_{\mathsf{OT}}$ functionality in the plain model. For our construction of 4-round OT protocol in the plain model, we prove security in the standalone setting.

2.4 Proof Systems

An n-round delayed-input interactive protocol for deciding a language $\mathcal{L}$ corresponding to a relation $\mathcal{R}$ is denoted by $\langle \mathsf{P}, \mathsf{V} \rangle$ and it proceeds as follows:

- At the beginning of the protocol, P and V receive the size of the instance and execute the first $n-1$ rounds.
- At the start of the last round, P receives input $(x, w) \in \mathcal{R}$ and V receives x. Upon receiving the last round message from P, V outputs 0 or 1.

For our protocols, we rely on ZK arguments and WI for NP that satisfy delayed-input soundness. Fix any language $\mathcal{L}$. Let $\langle \mathsf{P}, \mathsf{V} \rangle$ denote the execution of a protocol between a PPT prover P and a verifier V, let $\mathsf{V}_{\mathsf{out}}$ denote the output of the verifier and let $\mathcal{V}_{\mathcal{A}}\langle \mathsf{P}, \mathsf{V} \rangle$ denote the transcript together with the state and randomness of a party $\mathcal{A} \in \{\mathsf{P}, \mathsf{V}\}$ at the end of an execution of a protocol.

Definition 4. (Delayed-input Zero Knowledge Argument). *For any fixed language $\mathcal{L}$, we say $\langle P, V \rangle$ is a delayed-input zero knowledge argument system ZK for $\mathcal{L}$ if the following properties hold:*

- *Completeness: For all $x \in \mathcal{L}$,*
$$\Pr[V_{out}\langle P, V \rangle = 1] = 1 - neg(\kappa),$$
where the probability is over the random coins of P and V.
- *Adaptive Soundness: For all polynomial size P^* and all $x \notin \mathcal{L}$ sampled by P^* adaptively depending upon the first $n-1$ rounds,*
$$\Pr[V_{out}\langle P^*, V \rangle = 1] = neg(\kappa).$$
- *Zero Knowledge: There exists a PPT simulator $\mathcal{S}$ such that for all PPT V^* and all $x \in \mathcal{L}$,*
$$\left| \Pr[V^*(\mathcal{V}_{\mathcal{A}}\langle P(x, w), V^* \rangle) = 1] - \Pr[V^*(\mathcal{S}^{V^*}(x)) = 1] \right| = neg(\kappa).$$

Four round delayed-input statistical zero knowledge arguments can be obtained from [LS91] by relying on two round statistically hiding commitments.

Definition 5. (Delayed-input Witness Indistinguishability). *For any fixed language $\mathcal{L}$, we say $\langle P, V \rangle$ is a delayed-input witness indistinguishability proof system WI for $\mathcal{L}$ if the following properties hold:*

- *Completeness: For all $x \in \mathcal{L}$,*
$$\Pr[V_{out}\langle P, V \rangle = 1] = 1 - neg(\kappa),$$
where the probability is over the random coins of P and V.

- *Adaptive Soundness: For all P^* and all $x \notin \mathcal{L}$ sampled by P^* adaptively depending upon the first $n-1$ rounds,*
$$\Pr[\mathsf{V}_{out}\langle \mathsf{P}^*, \mathsf{V}\rangle = 1] = \mathsf{neg}(\kappa).$$
- *Witness Indistinguishability: For two valid witnesses w_1 and w_2, such that $(x_1, w_1) \in L$ and $(x_2, w_2) \in \mathcal{L}$, the following holds for all polynomial size V^* and all $x \in \mathcal{L}$,*
$$\left| \Pr[\mathsf{V}^*(\mathcal{V}_{\mathcal{A}}\langle \mathsf{P}(x, w_1), \mathsf{V}^*\rangle) = 1] - \Pr[\mathsf{V}^*(\mathcal{V}_{\mathcal{A}}\langle \mathsf{P}(x, w_2), \mathsf{V}^*\rangle) = 1] \right| = \mathsf{neg}(\kappa).$$

3 Round-Optimal UC-Secure OT from wU-EGA

In this section, we demonstrate how to construct a two-round UC-secure OT protocol in the CRS model based on any weak unpredictable effective group action (EGA) (Definition 2). For background material on EGA, see Section 2.2. For simplicity, we begin with a construction of two-round (round optimal) OT in the CRS model that is UC-secure against a *malicious* sender but only a *semi-honest* receiver. Subsequently, we show how to augment the construction in order to also achieve UC-security against a malicious receiver.

3.1 Warm-Up: 2-round UC-OT against Semi-Honest Receiver

We provide a brief overview of our protocol. The initial protocol is described as follows. The crs consists of two set elements $(x_0, x_1) = (g_0 \star x, g_1 \star x)$. The receiver has its input choice bit b. It constructs the OT receiver message z by sampling a random group element $r \leftarrow_R G$ as follows:

$$z = r \star x_b$$

The sender has input messages $(m_0, m_1) \in \{0,1\}^\kappa$. The sender uses z and the $\mathsf{crs} = (x_0, x_1)$ to compute the second OT message by sampling random group elements $k_0, k_1 \leftarrow_R G$ as follows:

$$y_0 = k_0 \star x_0, \quad \gamma_0 = H(k_0 \star z) \oplus \mathsf{m}_0,$$
$$y_1 = k_1 \star x_1, \quad \gamma_1 = H(k_1 \star z) \oplus \mathsf{m}_1.$$

The receiver uses the randomness r to decrypt m_b as follows:

$$m_b = \gamma_b \oplus H(r \star y_b).$$

Let td denote the trapdoor of the CRS as follows:

$$\mathsf{crs} = (g_0 \star x, g_1 \star x), \quad \mathsf{td} = g_1(g_0)^{-1},$$

The protocol is secure against a malicious sender since z perfectly hides b. If $b = 0$, then the honest receiver constructs $z = r \star x_0$. The same z can be opened to choice bit $b = 1$ with randomness r' (by using the trapdoor td) as follows:

$$z = r \star x_0 = r \cdot (g_1(g_0)^{-1}) \star x_1 = r' \star x_1 \text{ where } r' = rg_1(g_0)^{-1}.$$

Using the above observation, the simulator constructs $z = r \star x_0$ and extracts m_0 and m_1 using randomness r and r' respectively. Next, we argue security against a semi-honest receiver. We show that if the receiver computes m_{1-b} by querying $H(k_1 \star z)$ to the random oracle then one can build an adversary for breaking the weak unpredictability property. The details of our reduction can be found in Section. 3.1. Our reduction requires the knowledge of the receiver's randomness r to plug in the challenge instance of the weak unpredictability game into the sender's OT messages. Also, z perfectly hides b and as a result the simulator cannot extract the corrupt receiver's choice bit b during simulation. These are the reasons due to which the current construction only attains malicious security against a corrupt sender. Our construction and proof sketch follows.

The Construction. Let $(G, X, \star)$ be a wU-EGA with x being a publicly available element in the set X. Also let $H : X \to \{0,1\}^\ell$ be a hash function (modeled in the proof as a random oracle). Our construction is a tuple of four PPT algorithms $(\mathsf{Setup}, \mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g_0, g_1 \leftarrow_R G$ and output $\mathsf{crs} = (x_0, x_1)$ where
$$x_0 = g_0 \star x, \quad x_1 = g_1 \star x.$$
- $\mathsf{OTR}(\mathsf{crs}, b)$: Sample uniformly at random $r \leftarrow_R G$ and compute $z = r \star x_b$. Output the receiver message $\mathsf{ot}_1 = z$ and the receiver state $\mathsf{st} = (b, r)$.
- $\mathsf{OTS}(\mathsf{crs}, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: Parse $\mathsf{crs} = (x_0, x_1)$ and $\mathsf{ot}_1 = z$. Sample uniformly at random $k_0, k_1 \leftarrow_R G$ and output the sender message $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, where
$$y_0 = k_0 \star x_0, \quad \gamma_0 = H(k_0 \star z) \oplus \mathsf{m}_0,$$
$$y_1 = k_1 \star x_1, \quad \gamma_1 = H(k_1 \star z) \oplus \mathsf{m}_1.$$
- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (b, r)$ and $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, and output the recovered message as
$$\mathsf{m}' = \gamma_b \oplus H(r \star y_b).$$

Correctness. Correctness of the scheme follows by inspection.

Security. We state and prove the following theorem.

Theorem 4. *Assuming that $(G, X, \star)$ be a wU-EGA and H is a random oracle, the above construction implements the $\mathcal{F}_{\mathsf{OT}}$ functionality in the common reference string + random oracle model against a malicious sender and a semi-honest receiver.*

Security against Malicious Sender (Informal). Note that the receiver's choice bit b is hidden statistically. Also, note that z is in fact an equivocal commitment to b given the "discrete log" of x_1 w.r.t. x_0, i.e. the group element $g_1(g_0)^{-1}$. Hence, the simulator can generate a CRS-trapdoor pair $(\mathsf{crs}, \mathsf{td})$ as
$$\mathsf{crs} = (g_0 \star x, g_1 \star x), \quad \mathsf{td} = g_1(g_0)^{-1},$$
and recover both the sender messages m_0 and m_1.

Security against Semi-Honest Receiver (Informal). We will prove the following lemma:

Lemma 1. *Assuming that $(G, X, \star)$ be a wU-EGA and H is a random oracle, the above construction is UC-secure in the common reference string + random oracle model against a semi-honest receiver.*

Proof. Given an wU-EGA challenge of the form $(x, x^*, y = k \star x)$, the goal is to predict $y^* = k \star x^*$. Suppose $\mathcal{A}$ is an adversary that breaks OT security. We show that there exists an adversary $\mathcal{A}'$ for wu-EGA given $\mathcal{A}$. The reduction proceeds as follows (the reduction already knows the corrupt receiver's choice bit b and output m_b, and simulates hash function H as a random oracle):

- Simulate the CRS as $\mathsf{crs} = (x_0, x_1)$ where :
$$x_b = x^*, \quad x_{1-b} = x.$$

- On behalf of the receiver, sample $r \leftarrow_R G$ and compute $z = r \star x_b$. Output the receiver message $\mathsf{ot}_1 = z$.
- On behalf of the sender, sample $k' \leftarrow_R G$ and output simulated sender OT message as $\mathsf{ot}'_2 = (y_0, y_1, \gamma_0, \gamma_1)$ where

$$y_b = k' \star x_b, \quad \gamma_b = H(k' \star z) \oplus \mathsf{m}_b, y_{1-b} = y, \quad \gamma_{1-b} \leftarrow_R \{0,1\}^\ell.$$

Let E be the event that $\mathcal{A}$ queries the random oracle with input $k \star z$. Let us denote the real world (resp. simulated) OT sender message as ot_2 (resp. ot'_2). Then, we denote the advantage of a corrupt receiver breaking sender privacy as follows.

$$\begin{aligned}
&\big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1]\big| \\
&\quad = \big|(\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|E] \cdot \Pr[E] + \Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] \cdot \Pr[\overline{E}]) \\
&\qquad\quad -(\Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|E] \cdot \Pr[E] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}] \cdot \Pr[\overline{E}])\big| \\
&\quad = \big|(\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|E] \cdot \Pr[E] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|E] \cdot \Pr[E]) \\
&\qquad\quad +(\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] \cdot \Pr[\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}] \cdot \Pr[\overline{E}])\big| \\
&\quad = \big|\Pr[E] \cdot (\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|E] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|E]) \\
&\qquad\quad - \Pr[\overline{E}] \cdot (\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}])\big| \\
&\quad \leq \Pr[E] \cdot \big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|E] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|E]\big| \\
&\qquad\quad + \Pr[\overline{E}] \cdot \big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1\big|\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}]\big| \\
&\quad \leq \Pr[E] + \big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}]\big|.
\end{aligned}$$

where ot_2 is computed honestly following the honest sender algorithm and (m_0, m_1), and ot'_2 is computed as described above. The second last inequality follows due to triangle inequality. Rearranging the terms yields the following inequality :

$$\big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1]\big| - \big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}]\big| \leq \Pr[E]$$

Note that the simulation is perfect assuming event E does not occur, since H is a random oracle and since

$$y_{1-b} = y = k \star x = k \star x_{1-b}.$$

In such a case, an honestly computed γ_{1-b} is indistinguishable from a random γ_{1-b} if the adversary $\mathcal{A}$ does not query H on $k \star z$. This follows from the random oracle assumption. Thus the following occurs with negligible probability:

$$|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1|\overline{E}] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1|\overline{E}]| \leq \mathsf{neg}(\kappa).$$

This reduces the above equation to the following:

$$\big|\Pr[\mathcal{A}(\mathsf{ot}_2) \to 1] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1]\big| - \mathsf{neg}(\kappa) \leq \Pr[E]$$

Next, we construct our adversary $\mathcal{A}'$ for wU-EGA provided event E occurs, i.e. $\mathcal{A}$ queries H on $k \star z$. The adversary $\mathcal{A}$ distinguishes ot_2 and ot'_2 if it obtains information about m_{1-b}. Given the simulated ensemble,

$$(\mathsf{crs}, b, \mathsf{m}_b, \mathsf{ot}_1 = z, \mathsf{ot}'_2 = (y_0, y_1, \gamma_0, \gamma_1)),$$

if $\mathcal{A}$ manages to recover message m_{1-b} by querying (conditioned on occurrence of event E) the random oracle on $z^* = k \star z$, then the following holds true:

$$z^* = k \star z = k \star (r \star x_b) = r \star (k \star x_b) = r \star (k \star x^*) = r \star y^*.$$

Hence, the adversary $\mathcal{A}'$ recovers (with non-negligible probability)

$$y^* = r^{-1} \star z^*,$$

thereby violating the weak unpredictability of the EGA. Thus, the advantage of an adversary $\mathcal{A}'$ in the weak unpredictability game will be as follows:

$$\left| \Pr[\mathcal{A}(\mathsf{ot}_2) \to 1] - \Pr[\mathcal{A}(\mathsf{ot}'_2) \to 1] \right| \leq \Pr[E] \leq \Pr[\mathcal{A}' \text{ wins wU-EGA game}] \leq \mathsf{neg}(\kappa).$$

This completes the proof of Lemma 1 and, hence, the proof of Theorem 4. □

3.2 2-round Maliciously secure UC-OT

We now show how to augment the construction in order to also achieve UC-security against a malicious receiver. We add security against a malicious receiver by forcing the receiver to send a NIWI (proof of knowledge) π proving correct construction of its OT message corresponding to the following statement:

$$\exists b \in \{0,1\}, r \in G : z = r \star x_b$$

The sender verifies the proof as part of the OT protocol. The proof allows a simulator to extract the choice bit b and randomness r to complete reduction. The knowledge of r is required for the security reductions among the hybrids. The NIWI can be performed by applying Fiat-Shamir Transform on the Sigma protocols of [DG19].[5] This yields the first round optimal OT from weak unpredictability property and it can be instantiated based on computational CSIDH assumption.

Additional Requirement. Let $(G, X, \star)$ be a wU-EGA with x being a publicly available element in the set X. We denote the NIWI proof of knowledge (NIWI-POK) system as follows:

$$\mathsf{NIWI} = (\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify}),$$

that is capable of generating proofs for OR relations of the following form with respect to a tuple $(x_0, x_1, z) \in X \times X \times X$:

$$\exists r \in G : (z = r \star x_0) \vee (z = r \star x_1),$$

where the tuple (x_0, x_1, z) is the proof statement and the witness is a tuple of the form $(r, b) \in G \times \{0,1\}$. We describe the corresponding protocol next.

3.2.1 The NIWI Construction for 2 round UC-OT

We build upon [DG19] to construct a NIWI (non-interactive witness indistinguishable) OR proof system, wherein the simulator can always simulate the proof without programming the random oracle. For proof of knowledge, we rely on the well-known transformation from [Pas03] to obtain a poly-time straight-line NIWI proof of knowledge (NIWIpok), in the non-programmable RO model. A similar transformation was also used in [CJS14] paper to construct a NIWIpok in the non-programmable RO+CRS model. We refer to Sec. 2.4 for formal definitions.

[5]The recent work of [BDK+21] constructs a similar NIZK. But it is based on the decisional CSIDH assumption, and is hence insufficient for our purpose.

Construction. Let $(G, X, \star)$ be an EGA with $(x_0 = g_0 \star x, x_1 = g_1 \star x)$ being the crs. Let H be the random oracle. The prover and verifier has the input statement z. The prover possesses choice bit b and private randomness r such that $z = x \star x_b$. The prover proves the following statement:

$$\exists r \in G : (z = r \star x_0) \vee (z = r \star x_1)$$

We denote $1 - b$ as $\bar{b}$ for a bit $b \in \{0, 1\}$. The NIWI protocol is as follows:

- $\mathsf{Prove}((x_0, x_1, z), (r, b))$:
  Compute the first message of the Sigma protocol by repeating the following steps κ times for $i \in [\kappa]$:

  – Sample $s_{i,0}, s_{i,1} \leftarrow_R G$ and a bit $e_{i,\bar{b}} \leftarrow_R \{0, 1\}$.
  – Compute $w_{i,b} = s_{i,b} \star x_b$.
  – If $e_{i,\bar{b}} == 0$ then set $w_{i,\bar{b}} = s_{i,\bar{b}} \star x_{\bar{b}}$.
  – If $e_{i,\bar{b}} == 1$ then set $w_{i,\bar{b}} = s_{i,\bar{b}} \star z$.

  Compute responses to challenge bits 0 and 1 for each repetition and commit to the openings by repeating the following steps κ times for $i \in [\kappa]$:

  – Computing opening for challenge bit 0: Set $E^0_{i,b} = E^0_{i,\bar{b}} = e_{i,\bar{b}}$. Compute openings as $d^0_{i,0}, d^0_{i,1}$ where :
    – $d^0_{i,\bar{b}} = s_{i,\bar{b}}$
    – $d^0_{i,b} == s_{i,b}$ if $E^0_{i,b} == 0$.
    – $d^0_{i,b} == r^{-1}s_{i,b}$ if $E^0_{i,b} == 1$.

    Denote the openings as $\sigma^0_i = (E^0_{i,0}, E^0_{i,1}, d^0_{i,0}, d^0_{i,1})$. Commit to the openings as $C^0_i = H(\sigma^0_i)$.

  – Computing opening for challenge bit 1: Set $E^1_{i,\bar{b}} = e_{i,\bar{b}}$ and $E^1_{i,b} == 1 - e_{i,\bar{b}}$. Compute openings as $d^1_{i,0}, d^1_{i,1}$ where :
    – $d^1_{i,\bar{b}} = s_{i,\bar{b}}$
    – $d^1_{i,b} == s_{i,b}$ if $E^1_{i,b} == 0$.
    – $d^1_{i,b} == r^{-1}s_{i,b}$ if $E^1_{i,b} == 1$.

    Denote the openings as $\sigma^1_i = (E^1_{i,0}, E^1_{i,1}, d^1_{i,0}, d^1_{i,1})$. Commit to the openings as $C^1_i = H(\sigma^1_i)$.

  Compute the challenge string $\mathbf{c} = \{c_i\}_{i \in \kappa} = H(\{w_{i,0}, w_{i,1}, \mathsf{crs}, z, C^0_i, C^1_i\}_{i \in [\kappa]})$.

  Send the NIWI proof $\mathsf{pf} = \{w_{i,0}, w_{i,1}, C^0_i, C^1_i, \sigma^{c_i}_i\}_{i \in [\kappa]}$.

- $\mathsf{Verify}((x_0, x_1, z), \mathsf{pf})$:
  Parse the proof as $\mathsf{pf} = \{w_{i,0}, w_{i,1}, C^0_i, C^1_i, \sigma_i\}_{i \in \kappa}$. Compute the challenge string $\mathbf{c} = \{c_i\}_{i \in \kappa} = H(\{w_{i,0}, w_{i,1}, \mathsf{crs}, z, C^0_i, C^1_i\}_{i \in [\kappa]})$. Repeat the following steps for $i \in [\kappa]$:

  – Abort if $C^{c_i}_i \neq H(\sigma_i)$.

- Parse the opening $\sigma_i = (E_{i,0}, E_{i,1}, d_{i,0}, d_{i,1})$.
- Abort if $E_{i,0} \oplus E_{i,1} \neq c_i$.
- For $\beta \in \{0,1\}$: Perform the following checks for the βth branch corresponding to challenge $E_{i,\beta}$:
    - If $E_{i,\beta} == 0$: Abort if $w_{i,\beta} \neq d_{i,\beta} \star x_\beta$.
    - If $E_{i,\beta} == 1$: Abort if $w_{i,\beta} \neq d_{i,\beta} \star z$.

The verifier accepts the proof if it has not aborted.

Proof of Knowledge. We assume the existence of two witness extractor algorithms $(\mathsf{Ext}_1, \mathsf{Ext}_2)$. The witness extractor Ext_1 relies on the observability of the random oracle to either uniquely extract the witness b or extract the randomness r of the prover. The extractor Ext_1 observes the random oracle queries on σ_i^0 and σ_i^1 to extract $(d_{i,0}^0, d_{i,0}^1)$ and $(d_{i,1}^0, d_{i,1}^1)$. For bit $\gamma \in \{0,1\}$, if $E_{i,\gamma}^0 \neq E_{i,\gamma}^1$ then the extractor sets the receiver's choice bit as $b = \gamma$ and extracts the randomness r from $d_{i,\gamma}^0$ and $d_{i,\gamma}^1$ such that $z = r \star x_\gamma$. If the extractor is able to repeat the same process for $\overline{\gamma}$ from a repetition $j \neq i \in [\kappa]$ then it would extract r' such that $z = r' \star x_{\overline{\gamma}}$. The knowledge of r and r' allows the extractor to compute the trapdoor g_2 of the crs such that $x_1 = g_2 \star x_0$. This breaks computational CSIDH assumption corresponding to the challenge instance (x, x_0, x_1). Otherwise, the extractor uniquely extracts the adversary's choice bit and randomness. The only other way the corrupt prover can construct an accepting proof and yet hamper witness extraction if it correctly predicts the challenge string $\mathbf{c}$. However, that occurs with negligible probability for κ sessions in the random oracle model. This follows from the witness extraction property of the [Pas03] transformation. The extractor Ext_2 rewinds the prover in the random oracle queries and programs the random oracle to return different challenge strings, which allows it to extract the randomness r from two pairs of accepting proofs. Looking ahead, this extractor will be used to argue indistinguishability between hybrids and perform security reductions. It will not be used in simulation and hence rewinding the prover does not hinder UC-security of the OT protocol.

Witness-Indistinguishability. The simulator correctly simulates the proof by setting $z = r \star x_0$ and $b = 0$ and running the honest prover algorithm. The first message $(w_{i,0}, w_{i,1},)$, the commitments (C_i^0, C_i^1) and opening σ^{c_i} hides (r, b) since the commitment $C_i^{\overline{c_i}}$ hides the openings $\sigma_i^{\overline{c_i}}$ in the random oracle model.

3.2.2 The 2-round Maliciously Secure UC-OT Construction

Let $(G, X, \star)$ be a wU-EGA with x being a publicly available element in the set X. Also let $H : X \to \{0,1\}^\ell$ be a hash function (modeled in the proof as a random oracle). Our construction is a collection of four PPT algorithms $(\mathsf{Setup}, \mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g_0, g_1 \leftarrow_R G$, and output $\mathsf{crs} = (x_0, x_1)$, where

$$x_0 = g_0 \star x, \quad x_1 = g_1 \star x.$$

- $\mathsf{OTR}(\mathsf{crs}, b)$: Sample uniformly at random $r \leftarrow_R G$ and compute $z = r \star x_b$. Output the receiver message $\mathsf{ot}_1 = (z, \pi)$ and the receiver state $\mathsf{st} = (b, r)$, where

$$\pi \leftarrow_R \mathsf{NIWI.Prove}((x_0, x_1, z), (r, b)).$$

- $\mathsf{OTS}(\mathsf{crs}, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: Parse $\mathsf{ot}_1 = (z,\pi)$ and proceed as follows:
  - If $\mathsf{NIWI.Verify}((x_0, x_1, z), \pi) = 0$, output $\perp$.
  - Otherwise, sample uniformly at random $k_0, k_1 \leftarrow_R G$ and output the sender message $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, where

$$y_0 = k_0 \star x_0, \quad \gamma_0 = H(k_0 \star z) \oplus \mathsf{m}_0,$$

$$y_1 = k_1 \star x_1, \quad \gamma_1 = H(k_1 \star z) \oplus \mathsf{m}_1.$$

- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (b, r)$ and $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, and output the recovered message as

$$\mathsf{m}' = \gamma_b \oplus (r \star y_b).$$

Correctness. Correctness of the scheme follows by inspection.

Security. The security of our protocol is summarized below.

Theorem 5. *Assuming that $(G, X, \star)$ is a wU-EGA,* NIWI *is a NIWI proof of knowledge, and H is a random oracle, the above construction implements the $\mathcal{F}_{\mathsf{OT}}$ functionality in the common reference string + random oracle model and it is UC-secure against malicious adversaries.*

Proof. At a high level, the proof is very similar to the proof for our semi-honest construction, with the additional guarantees provided by the (NIWI-POK) system allowing us to prove security against a malicious receiver. The detailed proof is presented below. We consider the corruption cases for a maliciously corrupt sender and a maliciously corrupt receiver as follows.

Security against Malicious Sender. We provide the simulation algorithm $\mathcal{S}_1$ against a corrupt sender as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g_0, g_1 \leftarrow_R G$ and output $\mathsf{crs} = (x_0, x_1)$, where

$$x_0 = g_0 \star x, \quad x_1 = g_1 \star x.$$

  Output the trapdoors of the crs as $\mathsf{td} = (g_0, g_1)$ to the simulator $\mathcal{S}_1$.

- $\mathsf{OTR}(\mathsf{crs})$: The simulator samples uniformly at random $r \leftarrow_R G$ and computes $z = r \star x_0$. It computes the simulated proof π as follows:

$$\pi \leftarrow_R \mathsf{NIWI.Prove}((x_0, x_1, z), (r, 0)).$$

  Outputs the simulated receiver message as $\mathsf{ot}_1 = (z, \pi)$ and the receiver state $\mathsf{st} = (0, r)$.

- $\mathsf{OTS}(\mathsf{crs}, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: The corrupt sender sends $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$.

- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (0, r)$, $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, setup trapdoor $\mathsf{td} = (g_0, g_1)$ and compute the recovered messages as

$$\mathsf{m}_0 = \gamma_0 \oplus (r \star y_0).$$

$$\mathsf{m}_1 = \gamma_1 \oplus (r g_1^{-1} g_0 \star y_0).$$

  It invokes the $\mathcal{F}_{\mathsf{OT}}$ functionality with sender's messages (m_0, m_1) and completes simulation.

We argue security against a corrupt sender as follows.

- $\texttt{Hyb}_0$: This is the real world execution of the protocol.

- $\texttt{Hyb}_1$: Same as $\texttt{Hyb}_0$, except the simulator constructs π for witness $(r, 0)$. Indistinguishability follows from the WI property of the NIWI protocol.

- $\texttt{Hyb}_2$: Same as $\texttt{Hyb}_1$, except the simulator sets $z = r \star x_0$ and extracts sender's input messages (m_0, m_1) following the simulation algorithm. The receiver's OT message - (z, π) perfectly hides the choice bit b between $\texttt{Hyb}_1$ and $\texttt{Hyb}_2$. $z = r \star x_0$ can also be opened to choice bit using randomness $rg_1^{-1}g_0$. This is the ideal world execution of the protocol.

Security against Malicious Receiver (Informal). We provide the simulation algorithm $\mathcal{S}_2$ against a corrupt receiver as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g_0, g_1 \leftarrow_R G$ and output $\mathsf{crs} = (x_0, x_1)$, where
$$x_0 = g_0 \star x, \quad x_1 = g_1 \star x.$$
Output the trapdoors of the crs as $\mathsf{td} = (g_0, g_1)$ to the simulator $\mathcal{S}_2$.

- $\mathsf{OTR}(\mathsf{crs})$: The receiver sends OT message as $\mathsf{ot}_1 = (z, \pi)$.

- $\mathsf{OTS}(\mathsf{crs}, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$:
  - $\mathcal{S}_2$ invokes $\mathsf{NIWI.Ext}_1(\pi)$ to get $b \in \{0, 1, \perp\}$. If $b = \perp$ then it aborts else $\mathcal{S}_2$ invokes $\mathcal{F}_{\mathsf{OT}}$ with input choice bit b to obtain m_b.
  - $\mathcal{S}_2$ samples uniformly at random $k_0, k_1 \leftarrow_R G$ and outputs the simulated sender message $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$, where
$$y_b = k_b \star x_b, \quad \gamma_b = H(k_b \star z) \oplus \mathsf{m}_b,$$
$$y_{\bar{b}} = k_{\bar{b}} \star x_{\bar{b}}, \quad \gamma_{\bar{b}} \leftarrow \{0, 1\}^{|m_{\bar{b}}|}$$

- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2)$: The corrupt receiver performs its own adversarial algorithm. $\mathcal{S}_2$ aborts if the corrupt receiver invokes the random oracle on $k_{\bar{b}} \star z$.

We argue security against a corrupt receiver as follows. The simulator extracts the receiver's choice bit from the NIWI proof and aborts if extraction fails. Soundness of the NIWI protocol ensures that extraction succeeds if the proof π verifies. CSIDH assumption ensures that a corrupt receiver cannot query $k_{\bar{b}} \star z$ to the random oracle H. We provide the formal hybrids and indistinguishability argument as follows:

- $\texttt{Hyb}_0$: This is the real world execution of the protocol.

- $\texttt{Hyb}_1$: Same as $\texttt{Hyb}_0$, except the simulator aborts if the NIWI extractor returns $b = \perp$. Indistinguishability follows from the correctness of extractor $\mathsf{NIWI.Ext}_1$ of the NIWI protocol. If an adversary distinguishes between the two hybrids by producing a proof π that verifies but not extractable, then the adversary for the NIWI protocol produces this proof as the response; hence breaking the soundness of the NIWI protocol.

– $\texttt{Hyb}_2$: Same as $\texttt{Hyb}_1$, except the simulator aborts if corrupt receiver queries the random oracle on $k_{\bar{b}} \star z$. We prove that the two hybrids are indistinguishable if $(G, X, \star)$ is weak unpredictable EGA. This is the ideal world execution of the protocol and it completes our simulation.

Lemma 2. *If $(G, X, \star)$ is a weak unpredictable EGA, then $\texttt{Hyb}_1$ and $\texttt{Hyb}_2$ are computationally indistinguishable to adversary $\mathcal{A}$.*

Proof Sketch. Suppose the reduction is given an weak unpredictable EGA challenge of the form

$$(x, x^*, y = k \star x),$$

and the goal is to predict $y^* = k \star x^*$. The reduction guesses the receiver's choice bit as $b' \leftarrow_R \{0, 1\}$ and it proceeds as follows :

- Sample uniformly at random $h \leftarrow_R G$ and set:

$$x_b = x^*, \quad x_{1-b} = h \star x.$$

  Simulate the CRS as $\mathsf{crs} = (x_0, x_1)$.

- Upon receiving receiver's message (z, π), the reduction extracts receiver's randomness (r, b) from π by running the extractor $\mathsf{NIWI.Ext}_2$ (which involves rewinding[6] the prover in the NIWI protocol). It aborts the reduction if $b \neq b'$ else it invokes the $\mathcal{F}_{\mathsf{OT}}$ functionality with $b == b'$ to get m_b.

- On behalf of the sender, sample $k' \leftarrow_R G$ and output $\mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)$ where

$$y_b = k' \star x_b, \quad \gamma_b = H(k' \star z) \oplus \mathsf{m}_b,$$

$$y_{1-b} = h \star y, \quad \gamma_{1-b} \leftarrow_R \{0, 1\}^{\ell}.$$

  Note that the simulation is perfect since H is a random oracle and since

$$y_{1-b} = h \star y = h \star (k \star x) = k \star x_{1-b}.$$

Now, if there exists an adversary $\mathcal{A}$ that, given the simulated ensemble

$$(\mathsf{crs}, b, \mathsf{m}_b, \mathsf{ot}_1 = (z, \pi), \mathsf{ot}_2 = (y_0, y_1, \gamma_0, \gamma_1)),$$

and given the ability to query H, manages to recover (with non-negligible probability) even one bit of information about the message m_{1-b}, it must query the random oracle H on the set element $z^* = k \star z$. Now observe that

$$z^* = k \star z = k \star (r \star x_b) = r \star (k \star x_b) = r \star (k \star x^*) = r \star y^*.$$

Hence, the reduction recovers (with non-negligible probability)

$$y^* = r^{-1} \star z^*,$$

thereby violating the weak unpredictability of the EGA.

This completes the proof of Theorem 5. □

[6]Rewinding occurs only in the security reduction and not in the simulation; hence it does not hamper UC-security of the OT protocol.

Instantiation from wU-REGA. We finally note that our constructions and proofs work in essentially the same way from a restricted EGA provided that we can sample group elements from a distribution that is *statistically* close to uniform over the group G while retaining the ability to efficiently compute the action. We note that this is plausibly the case with respect to the instantiation of restricted EGA from CSIDH and other similar isogeny-based assumptions. We refer the reader to [DG19, ADMP20] for more details.

Leveraging this observation and Theorem 3 together with Theorem 5, we get the following corollary.

Corollary 3. *If the computational CSIDH assumption holds and if H is a random oracle, there exists a 2-round OT protocol that implements the $\mathcal{F}_{\mathsf{OT}}$ functionality in the common reference string + random oracle model and achieves UC-security against malicious adversaries.*

4 Round Optimal OT in Plain Model from wU-EGA

In this section we construct our round optimal OT with simulation-based security in the plain model from wU-EGA assumption.

4.1 Overview

We build upon the two round semi-honest OT protocol from Sec. 3.1. It can be observed that the receiver's choice bit b is perfectly hidden in the receiver OT message $\mathsf{ot}_1 = z$ (computed using randomness $g \in G$), even if the OT parameters (x_0, x_1) are generated by a malicious sender. We need to extract the receiver's choice bit and randomness to enable simulation security against a corrupt receiver. We rely on a three round WI proof of knowledge (denoted as WI) for this purpose, where the receiver proves that for statement (x_0, x_1, z) and witness (g, b) the following holds true:

$$\mathcal{C}_1((x_0, x_1, z), (g, b)) = 1, \qquad \text{iff } z = g \star x_b.$$

We require the WI proof system to be input-delayed where only the last message of the WI proof system depends on the statement being proven. We refer to Sec. 2.4 for formal definitions. In our protocol the receiver sends the first message π_1^{WI} of the proof in the first round, the sender sends the OT parameters (x_0, x_1) and the second round message π_2^{WI} of the proof in the second round, the receiver computes z and the final round message π_3^{WI} of the proof as the third OT message and the sender verifies the proof and sends $(y_0, y_1, \gamma_0, \gamma_1)$ as the final OT message. The receiver uses (g, b) to decrypt m_b. The simulator against a corrupt receiver invokes the witness extractor of WI to extract (g, b). The knowledge of g also allows us to break wU-EGA assumption when a malicious receiver computes both $(\mathsf{m}_0, \mathsf{m}_1)$. Meanwhile, receiver privacy follows the witness indistinguishability of the proof system. For every z, there always exists g_0 and g_1 such that $z = g_0 \star x_0 = g_1 \star x_1$.

Next, we need to extract a corrupt sender's input messages $(\mathsf{m}_0, \mathsf{m}_1)$ from $(y_0, y_1, \gamma_0, \gamma_1)$ to enable simulation security against a corrupt sender. We rely on a four round ZK proof of knowledge (denoted as ZK) for this purpose, where the sender proves that for statement (x, x_0, x_1) and witness (g_0, g_1) the following holds true:

$$\mathcal{C}_2((x_0, x_1), (g_0, g_1)) = 1, \qquad \text{iff } x_0 = g_0 \star x, x_1 = g_1 \star x.$$

We require the ZK proof system to be input-delayed where only the last message of the WI proof system depends on the statement being proven. We refer to Sec. 2.4 for formal definitions. In our protocol the receiver sends the first message π_1^{ZK} of the proof along with π_1^{WI} in the first round, the sender sends the OT parameters (x_0, x_1), the second round message π_2^{ZK} of the proof and $\pi_2\mathsf{WI}$ in the second round, the receiver computes z and π_3^{WI} and the third round message π_3^{ZK} of the proof as the third OT message and the sender verifies the WI proof, computes the final round message of ZK proof as π_4^{ZK} and sends $(y_0, y_1, \gamma_0, \gamma_1, \pi_4^{\mathsf{ZK}})$ as the final OT message. The receiver verifies the ZK proof and then computes the output. The simulator against a corrupt sender invokes the witness extractor of ZK to extract (g_0, g_1) and compute $(\mathsf{m}_0, \mathsf{m}_1)$. Meanwhile, the simulator against a corrupt receiver uses the ZK simulator to simulate the ZK proof.

The three round input-delayed WI proof system can be obtained[PRS02, KM20, BPS22] from non-interactive commitment schemes using the protocol of [FLS99]. The commitment scheme can be obtained from wU-EGA assumption via injective trapdoor one way function. The four round input-delayed ZK proof system can be constructed [PRS02, KM20, BPS22] from two-round statistically hiding commitment scheme which in turn can be constructed[7] from wU-EGA. As a result, we obtain the *first* round-optimal OT in plain model from wU-EGA which satisfies simulation security. Formal details of the protocol follows.

4.2 Our Protocol

Let $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ be a three round delayed input WI proof of knowledge for the following language $\mathcal{L}_1$ consisting of statement (x_0, x_1, z), witness (g, b) and NP verification circuit $\mathcal{C}_1$ described as follows, where $x_0, x_1, z \in X, g \in G, b \in \{0, 1\}$.

$$\begin{aligned} \mathcal{C}_1((x_0, x_1, z), (g, b)) &= 1, \quad \text{if } z = g \star x_b \\ &= 0, \quad \text{otherwise} \end{aligned}$$

Let $\mathsf{ZK} = (\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3, \mathsf{ZK}_4, \mathsf{ZK}_5)$ be a four round delayed input ZK proof of knowledge for the following language $\mathcal{L}_2$ consisting of statement (x, x_0, x_1), witness (g_0, g_1) and NP verification circuit $\mathcal{C}_2$ described as follows, where $x, x_0, x_1 \in X, g_0, g_1 \in G$.

$$\begin{aligned} \mathcal{C}_2((x, x_0, x_1), (g_0, g_1)) &= 1, \quad \text{if } x_0 = g_0 \star x, x_1 = g_1 \star x \\ &= 0, \quad \text{otherwise} \end{aligned}$$

Receiver has choice bit $b \in \{0, 1\}$. Sender has input bit-messages $(m_0, m_1) \in \{0, 1\}$. x is a public set element. $H : X \to \{0, 1\}$ is the Goldreich-Levin hash function. We describe our OT protocol as follows:

- $\mathsf{OTR}_1(1^\kappa, b)$: The receiver performs the following:
  - Runs the first round of WI on the security parameter to obtain $(\pi_1^{\mathsf{WI}}, \mathsf{st}_\mathsf{R}^{\mathsf{WI}}) \leftarrow \mathsf{WI}_1(1^\kappa, \mathcal{C}_1)$ for $\mathcal{L}_1$ with NP-verification circuit $\mathcal{C}_1$.
  - Runs the first round of ZK on the security parameter to obtain $(\pi_1^{\mathsf{ZK}}, \mathsf{st}_\mathsf{R}^{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^\kappa, \mathcal{C}_2)$ for $\mathcal{L}_2$ with NP-verification circuit $\mathcal{C}_2$.
  - Sends $\mathsf{ot}_1 = (\pi_1^{\mathsf{WI}}, \pi_1^{\mathsf{ZK}})$ as the first OT message and saves $\mathsf{st}_\mathsf{R} = (b, \mathsf{st}_\mathsf{R}^{\mathsf{WI}}, \mathsf{st}_\mathsf{R}^{\mathsf{ZK}})$ as the internal receiver state.

[7]The verifier sends (x_0, x_1) as the first round message by sampling $g_0, g_1 \leftarrow_R G$ and computing $x_0 = g_0 \star x, x_1 = g_1 \star x$. The committer commits to bit b by sampling g and computing the commitment as $z = g \star x_b$. The decommitment is (g, b). Bit b remains perfectly hidden. Binding follows from wU-EGA assumption since openings $(s_0, 0)$ and $(s_1, 1)$ for bits 0 and 1 help to find $r = s_0 \cdot s_1^{-1}$ such that $x_1 = r \star x_0$.

- $\mathsf{OTS}_1(1^\kappa, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: The sender computes the following:
  - Samples $g_0, g_1 \leftarrow_R G$ and computes the OT parameters as $x_0 = g_0 \star x$ and $x_1 = g_1 \star x$.
  - Computes second message of WI as $(\pi_2^{\mathsf{WI}}, \mathsf{st}_{\mathsf{S}}^{\mathsf{WI}}) \leftarrow \mathsf{WI}_2(1^\kappa, \mathcal{C}_1, \pi_1^{\mathsf{WI}})$.
  - Computes second message of ZK as $(\pi_2^{\mathsf{ZK}}, \mathsf{st}_{\mathsf{S}}^{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_2(1^\kappa, \mathcal{C}_2, \pi_1^{\mathsf{ZK}})$.
  - Sends $\mathsf{ot}_2 = \big(x_0, x_1, \pi_2^{\mathsf{WI}}, \pi_2^{\mathsf{ZK}}\big)$ as the second OT message and it stores

$$\mathsf{st}_{\mathsf{S}} = (\mathsf{m}_0, \mathsf{m}_1, x_0, x_1, \mathsf{ot}_1, \mathsf{st}_{\mathsf{S}}^{\mathsf{WI}}, \mathsf{st}_{\mathsf{S}}^{\mathsf{ZK}})$$

    as the internal sender state.

- $\mathsf{OTR}_2(\mathsf{st}_{\mathsf{R}}, \mathsf{ot}_2)$: The receiver does the following:
  - Samples $g \leftarrow_R G$ and computes $z = g \star x_b$.
  - Compute third message of WI as $\pi_3^{\mathsf{WI}} \leftarrow \mathsf{WI}_3((x_0, x_1, z), (g, b), \mathsf{st}_{\mathsf{R}}^{\mathsf{WI}}, \pi_2^{\mathsf{WI}})$ corresponding to statement (x_0, x_1, z) and witness (g, b).
  - Compute third message of ZK as $(\pi_3^{\mathsf{ZK}}, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_3(\mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}, \pi_2^{\mathsf{ZK}})$.
  - Sends the third OT message $\mathsf{ot}_3 = (z, \pi_3^{\mathsf{WI}}, \pi_3^{\mathsf{ZK}})$ and updates its internal state as $\mathsf{st}_{\mathsf{R}} = (b, g, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}})$.

- $\mathsf{OTS}_2(\mathsf{st}_{\mathsf{S}}, \mathsf{ot}_3)$: The sender computes the following:
  - The sender aborts if the WI proof fails to verify on statement (x_0, x_1, z), i.e.

$$\mathsf{WI}_4((x_0, x_1, z), \mathsf{st}_{\mathsf{S}}^{\mathsf{WI}}, \pi_3^{\mathsf{WI}}) = 0.$$

  - The sender computes the fourth message of ZK as $\pi_4^{\mathsf{ZK}} \leftarrow \mathsf{ZK}_3((x, x_0, x_1), (g_0, g_1), \mathsf{st}_{\mathsf{S}}^{\mathsf{ZK}}, \pi_3^{\mathsf{ZK}})$ corresponding to statement (x, x_0, x_1) and witness (g_0, g_1).
  - Sample uniformly at random $k_0, k_1 \leftarrow_R G$ and compute $(y_0, y_1, \gamma_0, \gamma_1)$, where

$$y_0 = k_0 \star x_0, \quad \gamma_0 = H(k_0 \star z) \oplus \mathsf{m}_0,$$

$$y_1 = k_1 \star x_1, \quad \gamma_1 = H(k_1 \star z) \oplus \mathsf{m}_1.$$

  - The sender sends fourth OT message $\mathsf{ot}_4 = (y_0, y_1, \gamma_0, \gamma_1, \pi_4^{\mathsf{ZK}})$ to the receiver.

- $\mathsf{OTD}(\mathsf{st}_{\mathsf{R}}, \mathsf{ot}_2)$: The receiver computes the following:
  - The receiver aborts if the ZK proof fails to verify on statement (x, x_0, x_1), i.e.

$$\mathsf{ZK}_5((x, x_0, x_1), \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}, \pi_4^{\mathsf{ZK}}) = 0.$$

  - The receiver parses $\mathsf{st}_{\mathsf{R}} = (b, g)$ and $\mathsf{ot}_4 = (y_0, y_1, \gamma_0, \gamma_1, \pi_4^{\mathsf{ZK}})$, and outputs the recovered message as m' where

$$\mathsf{m}' = \gamma_b \oplus H(r \star y_b).$$

We show that the above protocol provides indistinguishability based security against a malicious sender and simulation based security against a corrupt receiver by proving the following theorem.

Theorem 6. *Let $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ be a three round delayed input WI proof of knowledge for the following language $\mathcal{L}_1$, $\mathsf{ZK} = (\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3, \mathsf{ZK}_4, \mathsf{ZK}_5)$ be a four round delayed input ZK proof of knowledge for the following language $\mathcal{L}_2$, and $(G, X, \star)$ be a wU-EGA, then the above construction provides receiver privacy against a malicious sender and provides simulation-based security against a malicious receiver.*

Proof Overview. We first argue that our protocol satisfies simulation-based security against a corrupt sender and then we argue the same against a corrupt receiver.

Simulation against Corrupt Sender. Assume $x_1 = r \star x_0$. It can be observed that z perfectly hides b since for every $g_0 \in G$ there exists $g_1 = g_0 \cdot r^{-1}$ such that $z = g_0 \star x_0 = g_1 \star x_1$. When $b == 0$, the WI proof is constructed with the group element g_0 such that $z = g_0 \star x_0$. Meanwhile, when $b == 1$ the WI proof is constructed using g_1 as $z = g_1 \star x_1$ where g_0 and g_1 satisfies the above relation. A malicious sender distinguishing between a run of the OT protocol with receiver input choice bit $b = 0$ from a run of the OT protocol with receiver input choice bit $b = 1$ breaks the WI property of the proof system. Moreover, the simulator can extract both m_0 and m_1 given the trapdoors g_0 and g_1. The simulator obtains these trapdoors by invoking the ZK witness extractor algorithm $\mathsf{Ext}^{\mathsf{ZK}}$ on π^{ZK}.

Simulation against Corrupt Receiver. The simulator invokes the WI witness extractor algorithm, denoted as $\mathsf{Ext}^{\mathsf{WI}}$, to extract the witness (g, b) from the proof. The simulator invokes the $\mathcal{F}_{\mathsf{OT}}$ functionality with the extracted choice bit b to obtain m_b. The simulator constructs ot_4 with inputs $(\mathsf{m}_0, \mathsf{m}_1)$, where $m_{1-b} = 0$. The ZK proof is constructed by invoking the ZK simulator, denoted as $\mathcal{S}^{\mathsf{ZK}}$. An adversary breaks the security of the protocol if the WI proof is accepting and yet the witness extractor failed to extract a witness, or the corrupt receiver distinguishes the simulated ZK proof from a real one. In the later case, it breaks ZK property. In the former case, the corrupt receiver breaks the proof of knowledge property of the WI protocol. The other case, where the extractor extracts multiple valid witnesses also leads to an abort by the simulator. That event occurs when the receiver breaks the wU-EGA property.

Detailed Proof. We now present the detailed proof. We first show that our protocol satisfies simulation-based security against a corrupt sender and then we show the same against a corrupt receiver.

Simulation against Corrupt Sender. Assume $x_1 = r \star x_0$. It can be observed that z perfectly hides b since for every $g_0 \in G$ there exists $g_1 = g_0 \cdot r^{-1}$ such that $z = g_0 \star x_0 = g_1 \star x_1$. When $b == 0$, the WI proof is constructed with the group element g_0 such that $z = g_0 \star x_0$. Meanwhile, when $b == 1$ the WI proof is constructed using g_1 as $z = g_1 \star x_1$ where g_0 and g_1 satisfies the above relation. A malicious sender distinguishing between a run of the OT protocol with receiver input choice bit $b = 0$ from a run of the OT protocol with receiver input choice bit $b = 1$ breaks the WI property of the proof system. Moreover, the simulator can extract both m_0 and m_1 given the trapdoors g_0 and g_1. The simulator obtains these trapdoors by invoking the witness extractor algorithm $\mathsf{Ext}^{\mathsf{ZK}} = (\mathsf{Ext}_1^{\mathsf{ZK}}, \mathsf{Ext}_2^{\mathsf{ZK}}, \mathsf{Ext}_3^{\mathsf{ZK}})$ on the ZK proof. We present the simulation algorithm as follows:

- $\mathsf{OTR}_1(1^\kappa)$: The simulated receiver performs the following:
  - Runs the first round of WI on the security parameter to obtain $(\pi_1^{\mathsf{WI}}, \mathsf{st}_{\mathsf{R}}^{\mathsf{WI}}) \leftarrow \mathsf{WI}_1(1^\kappa, \mathcal{C}_1)$.
  - Computes first round of message of ZK by invoking the witness extractor as $(\pi_1^{\mathsf{ZK}}, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}) \leftarrow \mathsf{Ext}_1^{\mathsf{ZK}}(1^\kappa, \mathcal{C}_2)$.
  - Sends $\mathsf{ot}_1 = (\pi_1^{\mathsf{WI}}, \pi_1^{\mathsf{ZK}})$ as the first OT message and saves $\mathsf{st}_{\mathsf{R}} = (\mathsf{st}_{\mathsf{R}}^{\mathsf{WI}}, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}})$ as the internal receiver state.
- $\mathsf{OTS}_1(1^\kappa, (\mathsf{m}_0, \mathsf{m}_1), \mathsf{ot}_1)$: The corrupt sender sends $\mathsf{ot}_2 = \big(x_0, x_1, \pi_2^{\mathsf{WI}}, \pi_2^{\mathsf{ZK}}\big)$ as the second OT message.
- $\mathsf{OTR}_2(\mathsf{st}_{\mathsf{R}}, \mathsf{ot}_2)$: The simulated receiver does the following:
  - Samples $g \leftarrow_R G$ and computes $z = g \star x_0$.
  - Compute third message of WI as $\pi_3^{\mathsf{WI}} \leftarrow \mathsf{WI}_3((x_0, x_1, z), (g, 0), \mathsf{st}_{\mathsf{R}}^{\mathsf{WI}}, \pi_2^{\mathsf{WI}})$ corresponding to statement (x_0, x_1, z) and witness $(g, 0)$.
  - Computes third round message of ZK by invoking the witness extractor as $(\pi_3^{\mathsf{ZK}}, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}) \leftarrow \mathsf{Ext}_2^{\mathsf{ZK}}(\mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}}, \pi_2^{\mathsf{ZK}})$.
  - Sends the third OT message $\mathsf{ot}_3 = (z, \pi_3^{\mathsf{WI}}, \pi_3^{\mathsf{ZK}})$ and updates its internal state as $\mathsf{st}_{\mathsf{R}} = (0, g, \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}})$.
- $\mathsf{OTS}_2(\mathsf{st}_{\mathsf{S}}, \mathsf{ot}_3)$: The corrupt sender sends fourth round OT message $\mathsf{ot}_4 = (y_0, y_1, \gamma_0, \gamma_1, \pi_4^{\mathsf{ZK}})$ to the receiver.
- $\mathsf{OTD}(\mathsf{st}_{\mathsf{R}}, \mathsf{ot}_2)$: The simulated receiver computes the following:
  - Extract witness (g_0, g_1) by invoking the ZK witness extractor as $(g_0, g_1) \leftarrow \mathsf{Ext}_3^{\mathsf{ZK}}((x, x_0, x_1), \mathsf{st}_{\mathsf{R}}^{\mathsf{ZK}})$. The simulator aborts if extraction fails.
  - The simulator computes $(\mathsf{m}_0, \mathsf{m}_1)$ as follows:

$$\mathsf{m}_0 = \gamma_0 \oplus H(g \star y_0), \quad \mathsf{m}_1 = \gamma_1 \oplus H((g \cdot g_1^{-1} \cdot g_0) \star y_1)$$

  The simulator invokes $\mathcal{F}_{\mathsf{OT}}$ with $(\mathsf{m}_0, \mathsf{m}_1)$ to complete simulation.

Next, we provide the hybrids and the indistinguishability argument as follows:

- $\mathtt{Hyb}_0$: This is the real world execution of the protocol with the receiver's input as b.
- $\mathtt{Hyb}_1$: Same as $\mathtt{Hyb}_0$, except $(\pi_1^{\mathsf{ZK}}, \pi_3^{\mathsf{ZK}})$ are computed by invoking the ZK witness extractor algorithm $\mathsf{Ext}^{\mathsf{ZK}}$ and the simulator aborts if the extractor fails to extract a valid witness (g_0, g_1) from the ZK proof.

  Indistinguishability follows due to the correctness of the witness extraction algorithm of ZK proof system.

- $\mathtt{Hyb}_2$: This is same as $\mathtt{Hyb}_1$, except the simulator always sets $z = g \star x_0$ and the WI proof is constructed using the witness $(g, 0)$. The simulator also extracts $(\mathsf{m}_0, \mathsf{m}_1)$ following the simulation algorithm. This is the ideal world execution of the protocol.

  The adversary distinguishes between the two hybrids if it successfully breaks the WI property of the WI proof system. To simulate the reduction, the simulator considers $(g, 0)$ and $(g \cdot g_1^{-1} g_0, 1)$ as the two witnesses for the statement (x_0, x_1, z) corresponding to NP verification circuit $\mathcal{C}_2$. The WI challenger returns a proof which is returned to the adversary. The adversary distinguishing between the $\mathtt{Hyb}_1$ and $\mathtt{Hyb}_2$ is used to distinguish between the case where $(g \cdot g_1^{-1} g_0, 1)$ is used as the witness from the case where $(g, 0)$ is used as the witness. Note that z perfectly hides the (g, b).

Simulation against Corrupt Receiver. The simulator invokes the WI witness extractor algorithm, denoted as $\mathsf{Ext}^{\mathsf{WI}} = (\mathsf{Ext}_1^{\mathsf{WI}}, \mathsf{Ext}_2^{\mathsf{WI}})$, to extract the witness (g, b) from the proof. The simulator invokes the $\mathcal{F}_{\mathsf{OT}}$ functionality with the extracted choice bit b to obtain m_b. The simulator constructs ot_4 with inputs $(\mathsf{m}_0, \mathsf{m}_1)$, where $m_{1-b} = 0$. The ZK proof is constructed by invoking the ZK simulator, denoted as $\mathcal{S}^{\mathsf{ZK}} = (\mathcal{S}_1^{\mathsf{ZK}}, \mathcal{S}_2^{\mathsf{ZK}})$. An adversary breaks the security of the protocol if the WI proof is accepting and yet the witness extractor failed to extract a witness, or the corrupt receiver distinguishes the simulated ZK proof from a real one. In the later case, it breaks ZK property. In the former case, the corrupt receiver breaks the proof of knowledge property of the WI protocol. The other case, where the extractor extracts multiple valid witnesses also leads to an abort by the simulator. That event occurs when the receiver breaks the wU-EGA property. We present the simulation algorithm as follows:

- $\mathsf{OTR}_1(1^\kappa, b)$: The corrupt receiver sends $\mathsf{ot}_1 = (\pi_1^{\mathsf{WI}}, \pi_1^{\mathsf{ZK}})$ as the first OT message.

- $\mathsf{OTS}_1(1^\kappa, \mathsf{ot}_1)$: The simulated sender computes the following:

  - Samples $g_0, g_1 \leftarrow_R G$ and computes the OT parameters as $x_0 = g_0 \star x$ and $x_1 = g_1 \star x$.
  - Computes second message of WI by invoking the witness extractor as $(\pi_2^{\mathsf{WI}}, \mathsf{st}_\mathsf{S}^{\mathsf{WI}}) \leftarrow \mathsf{Ext}_1^{\mathsf{WI}}(1^\kappa, \mathcal{C}_1, \pi_1^{\mathsf{WI}})$.
  - Computes second message of ZK by invoking the Zero Knowledge simulator as $(\pi_2^{\mathsf{ZK}}, \mathsf{st}_\mathsf{S}^{\mathsf{ZK}}) \leftarrow \mathcal{S}_1^{\mathsf{ZK}}(1^\kappa, \mathcal{C}_2, \pi_1^{\mathsf{ZK}})$.
  - Sends $\mathsf{ot}_2 = \left(x_0, x_1, \pi_2^{\mathsf{WI}}, \pi_2^{\mathsf{ZK}}\right)$ as the second OT message and it stores st_S=(x_0, x_1, ot_1, $\mathsf{st}_\mathsf{S}^{\mathsf{WI}}$, $\mathsf{st}_\mathsf{S}^{\mathsf{ZK}}$) as the simulated sender state.

- $\mathsf{OTR}_2(\mathsf{ot}_2)$: The corrupt receiver sends the third OT message as $\mathsf{ot}_3 = (z, \pi_3^{\mathsf{WI}}, \pi_3^{\mathsf{ZK}})$.

- $\mathsf{OTS}_2(\mathsf{st}_\mathsf{S}, \mathsf{ot}_3)$: The simulated sender computes the following:

  - The simulator extracts the witness (g, b) by invoking the WI extractor as $(g||b) = \mathsf{Ext}_2^{\mathsf{WI}}((x_0, x_1, z), \mathsf{st}_\mathsf{S}^{\mathsf{WI}}, \pi_3^{\mathsf{WI}})$. The sender aborts if the extractor fails to extract a single witness, or the extractor extracts two valid witnesses- $(g_0, 0)$ and $(g_1, 1)$.
  - The simulator computes fourth message of ZK by invoking the Zero Knowledge simulator as $\pi_4^{\mathsf{ZK}} \leftarrow \mathcal{S}_2^{\mathsf{ZK}}((x, x_0, x_1), \mathsf{st}_\mathsf{S}^{\mathsf{ZK}}, \pi_3^{\mathsf{ZK}})$.
  - The simulator invokes $\mathcal{F}_{\mathsf{OT}}$ functionality with extracted input choice bit b to obtain m_b. The simulator computes $(y_0, y_1, \gamma_0, \gamma_1)$ with sender input message $(\mathsf{m}_0, \mathsf{m}_1)$ where $\mathsf{m}_{1-b} = 0$.

– The sender sends fourth OT message $\mathsf{ot}_4 = (y_0, y_1, \gamma_0, \gamma_1, \pi_4^{\mathsf{ZK}})$ to the receiver and concludes the simulation.

Next, we provide the hybrids and the indistinguishability argument as follows:

- $\mathtt{Hyb}_0$: This is the real world execution of the protocol with the sender's input as $(\mathsf{m}_0, \mathsf{m}_1)$.
- $\mathtt{Hyb}_1$: Same as $\mathtt{Hyb}_0$, except $(\pi_2^{\mathsf{ZK}}, \pi_4^{\mathsf{ZK}})$ are computed by invoking the ZK simulation algorithm $\mathcal{S}^{\mathsf{ZK}}$.

  Indistinguishability follows due to the zero knowledge property of the proof system.
- $\mathtt{Hyb}_2$: This is same as $\mathtt{Hyb}_1$, except the simulator aborts if the witness extractor fails to extract a single witness.

  The adversary distinguishes between the two hybrids if it constructs an accepting proof on which the witness extractor fails to extract. This breaks the WI proof of knowledge property where the proof is the adversarial response to the proof of knowledge challenger.
- $\mathtt{Hyb}_3$: This is same as $\mathtt{Hyb}_2$, except the simulator aborts if the witness extractor extracts two valid witnesses $(g_0, 0)$ and $(g_1, 1)$.

  The adversary distinguishing between the two hybrids can be used to break the one-wayness of the group action. The adversary outputting $(g_0, 0)$ and $(g_1, 1)$ on challenger input (x_0, x_1) can be used to find $r = g_0 \cdot g_1^{-1}$ such that $x_1 = r \star x_0$.
- $\mathtt{Hyb}_4$: Same as $\mathtt{Hyb}_3$, except the simulator extracts an unique witness (g, b) from the witness extractor algorithm, invokes $\mathcal{F}_{\mathsf{OT}}$ functionality with extracted input choice bit b to obtain m_b. The simulator computes ot_4 with sender input message $(\mathsf{m}_0, \mathsf{m}_1)$ where $\mathsf{m}_{1-b} = 0$. This is the ideal world execution of the protocol.

  An adversary distinguishing between the two hybrids can be used to break the wU-EGA property of $(G, X, \star)$. The reduction is similar to the proof of Lemma. 1, with the random oracle being replaced by the Goldreich-Levin hash function H.

Realization from wU-EGA. The three round input-delayed WI proof system can be obtained [PRS02, KM20, BPS22] from non-interactive commitment schemes using the protocol of [FLS99]. The commitment scheme can be obtained from wU-EGA assumption via injective trapdoor one way function. The four round input-delayed ZK proof system can be constructed [PRS02, KM20, BPS22] from two-round statistically hiding commitment scheme which in turn can be constructed from wU-EGA. As a result, we obtain the *first* round-optimal OT in plain model from wU-EGA which satisfies simulation security. Our result is summarized in Thm. 7.

Theorem 7. *Assuming $(G, X, \star)$ is a wU-EGA, there exists a four-round oblivious transfer protocol in the plain model that provides simulation based security against malicious corruptions of the parties.*

5 OT Extension from Reciprocal EGA

In this section, we discuss our three round OT extension protocol following a roadmap of observations. The maliciously secure OT protocol in [LGdSG21] fails to achieve UC security in three rounds, and would require four rounds.[8] However, their construction relies on an efficient two round semi-honest OT protocol. We observe that this semi-honest protocol can be used to implement a batch of $\ell = \mathcal{O}(\kappa)$ OTs, satisfying malicious security notions which are weaker than UC-security. This semi-honest to malicious security transformation requires a few additional checks, incurring $\mathcal{O}(1)$ cheap symmetric operations per OT. Finally, we show that this weaker notion of malicious security suffices for [KOS15] OT extension by applying the result of [CSW20]. We begin by introducing some additional definitions and notations surrounding EGA and REGA.

5.1 Reciprocal EGA and Reciprocal CSIDH

The OT protocol of Lai et al. [LGdSG21] is based on the reciprocal CSIDH assumption. This assumption is known to be quantum-equivalent to the computational CSIDH assumption, and does not have an analogue in the Diffie-Hellman setting. The construction of Lai et al. relies on crucially on the *quadratic twist* of an elliptic curve, which can be computed efficiently in the CSIDH setting. In this section, we present an abstraction of the quadratic twist and the reciprocal CSIDH assumption in the framework of (R)EGA. In particular, our abstraction captures all of the properties of quadratic twist and its associated hardness assumptions used by Lai et a.l (see [LGdSG21] for more details on the quadratic twist and its efficient computation in the CSIDH setting).

The Twist Map. Let $(G, X, \star)$ be an EGA (equivalently an REGA) as described above. We define a "twist" as a map $\mathcal{T} : X \to X$ that satisfies the following properties:

- For any $g \in G$ and any $x \in X$ we have $\mathcal{T}(g \star x) = g^{-1} \star \mathcal{T}(x)$.
- For any $x \in X$ and any *uniform* $g \leftarrow_R G$, we have: $g \star x \approx_s \mathcal{T}(g \star x)$.
- There exists a "twist-invariant" element $x_0 \in X$ such that $\mathcal{T}(x_0) = x_0$.

The Reciprocal EGA Assumption. Given an EGA $(G, X, \star)$, we say that the reciprocal assumption holds if for any security parameter $\kappa \in \mathbb{N}$ and for any PPT adversary $\mathcal{A}$, the following holds with overwhelmingly large probability:

$$\Pr[\mathsf{Expt}^{\mathsf{recEGA}}(\kappa, \mathcal{A}) = 1] < \mathsf{negl}(\kappa),$$

where the experiment $\mathsf{Expt}^{\mathsf{recEGA}}(\kappa, \mathcal{A})$ is as defined in Figure 3.

Remark 3. *We can similarly define a reciprocal REGA assumption where, in the corresponding experiment, all group elements (more concretely, the group elements g and s) are sampled from a distribution that is statistically close to uniform over the group G.*

Finally, we import the following theorem from [LGdSG21].

Theorem 8. ([LGdSG21]). *Assuming that the reciprocal CSIDH assumption holds, there exists an REGA satisfying the reciprocal REGA assumption.*

5.2 The Ideal Functionality $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$

We present in Figure 4 the ideal functionality $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ for executing a batch of ℓ (sender) random oblivious transfer with selective failure.

[8]This was pointed out by the authors of [LGdSG21] in their Eurocrypt 2021 presentation.

Experiment $\mathsf{Expt}^{\mathsf{recEGA}}(\kappa, \mathcal{A})$:

1. The challenger generates the description of an EGA $(G, X, \star)$ along with the "twist" map $\mathcal{T} : X \to X$ and a special "twist-invariant" element $x_{\mathcal{T}} \in X$.

2. The challenger then samples $g \leftarrow_R G$, sets $x = g \star x_{\mathcal{T}}$, and provides to the adversary $\mathcal{A}$ the tuple $(G, X, \star, \mathcal{T}, x_{\mathcal{T}}, x)$.

3. The adversary $\mathcal{A}$ outputs an element $z \in X$.

4. The challenger samples $s \leftarrow_R G$ and provides to the adversary $\mathcal{A}$ the set element $y = s \star x$.

5. The adversary $\mathcal{A}$ eventually outputs a pair of set elements $(z_0, z_1) \in X \times X$.

6. Output 1 if $(z_0, z_1) = (s \star z, s^{-1} \star z)$. Output 0 otherwise.

Figure 3: The Reciprocal EGA Experiment

5.3 Some Observations on the $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ Functionality

Before presenting our protocol we discuss the observations made by CSW that allow optimizing the base OT protocol in KOS. CSW showed that the following weakening for the base OTs suffices.

- *Security for a batch of ℓ OTs:* The base OT protocols are run in a batch of $\ell = \mathcal{O}(\kappa) > 3\mu$ OTs together. Simulation based security should hold for non-aborting parties for the batch together. This is weaker than concurrent composition of ℓ OTs, which are individually proven to be simulation secure.

- *Allow selective failure attacks:* The receiver in the base OT protocols possess random choice bits. KOS observed that even if a corrupt sender launches a selective failure on $\mathcal{O}(\kappa)$ receiver's random choice bits, still it suffices for the OT extension protocol. The same observation was used by CSW.

- *Indistinguishability-based security for an aborting receiver:* The base OT protocols are run on random inputs. They need to satisfy simulation based security for a non-aborting receiver, i.e. the simulator needs to extract the input choice bits of a corrupt receiver only if a corrupt receiver does not abort. If the receiver aborts then the simulator does not need to extract the receiver's choice. Instead, the base OTs need to only satisfy sender privacy. This permits a three round OT protocol with receiver sending the first and last message. The receiver is allowed to abort/send a junk third round OT message after it has computed its OT output from the sender's OT message. In such a scenario only sender privacy is required and hence the maliciously secure three round OT protocol of [LGdSG21] suffices for the base OT protocol even if it does not satisfy UC-security.

The first two properties are captured by CSW using the $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality which captures a batch of ℓ OT protocols with selective failure. As per the third property, the OT protocol implementing $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality needs to only provide indistinguishability based security for an aborting receiver. In what follows, we show how to efficiently construct an OT protocol implementing this ideal functionality from reciprocal EGA.

Figure 4: The ideal functionality $\mathcal{F}_{\textsf{SF-ROT}}$ for executing a batch of ℓ (Sender) Random Oblivious Transfer with Selective Failure

$\mathcal{F}_{\textsf{SF-ROT}}$

$\mathcal{F}_{\textsf{SF-ROT}}$ interacts with a sender S and receiver R for $i \in [\ell]$:

- On input $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid}, \mathbf{b})$ from R , parse $\mathbf{b} = \{b_i\}_{i\in[\ell]}$ where $b_i \in \{0,1\}$, perform the following for $i \in [\ell]$:
  - If R is corrupt then store $(\mathsf{rec}, \mathsf{sid}, i, b_i)$ and send $(\mathsf{rec}, \mathsf{sid}, i)$ to S.
  - If R is honest then sample $b_i \leftarrow \{0,1\}$ and then store $(\mathsf{rec}, \mathsf{sid}, i, b_i)$ and send $(\mathsf{rec}, \mathsf{sid}, i)$ to S.

  If a message of the form $(\mathsf{sen}, \mathsf{sid}, i, a_{0,i}, a_{1,i})$ is stored, send $(\mathsf{sent}, \mathsf{sid}, i, a_{b_i,i})$ to R and $(\mathsf{sent}, \mathsf{sid}, i, (a_{0,i}, a_{1,i}))$ to S for $i \in [\ell]$. Ignore future messages with same sid.
- On input $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid}, (\mathbf{a}_0, \mathbf{a}_1))$ from S where $\mathbf{a}_0, \mathbf{a}_1 \in \{0,1\}^{\ell\times\kappa} \cup \perp$, parse $\mathbf{a}_0 = \{a_{0,i}\}_{i\in[\ell]}$, $\mathbf{a}_1 = \{a_{1,i}\}_{i\in[\ell]}$, and perform the following for $i \in [\ell]$:
  - If S is corrupt then store $(\mathsf{sen}, \mathsf{sid}, i, (a_{0,i}, a_{1,i}))$ in memory and send $(\textsc{Received}, \mathsf{sid}, i)$ to R and S for $i \in [\ell]$.
  - If S is honest then sample $a_{0,i}, a_{1,i} \leftarrow \{0,1\}^\kappa$, store $(\mathsf{sen}, \mathsf{sid}, i, (a_{0,i}, a_{1,i}))$ in memory and send $(\textsc{Received}, \mathsf{sid}, i)$ to R and S.

  If a message of the form $(\mathsf{rec}, \mathsf{sid}, i, b_i)$ is stored, send $(\mathsf{sent}, \mathsf{sid}, i, (b_i, a_{b_i,i}))$ to R and $(\mathsf{sent}, \mathsf{sid}, i, (a_{0,i}, a_{1,i}))$ to S for $i \in [\ell]$. Ignore future messages with same sid.
- On input $(\textsc{Guess}, \mathsf{sen}, \mathsf{sid}, i, b')$ from a corrupt S, if $(\mathsf{rec}, \mathsf{sid}, i, b)$ exists in memory, $b' \in \{0,1,\perp\}$ and there does not exist $(\mathsf{sen}, \mathsf{sid}, i, (\textsc{Guess}, \cdot))$ in memory then store $(\mathsf{sen}, \mathsf{sid}, i, (\textsc{Guess}, b'))$ in memory and perform the following:
  - If $b' = \perp$, send $(\textsc{Cheat-Detected}, \mathsf{S})$ to R and $(\textsc{Cheat-Detected})$ to S.
  - If $b' = b$, send $(\textsc{Cheat-Undetected})$ to S.
  - If $b' \neq b$, send $(\textsc{Cheat-Detected}, \mathsf{S})$ to R and $(\textsc{Cheat-Detected})$ to S.
- On input $(\textsc{Abort}, \mathsf{rec}, \mathsf{sid})$ from R or $(\textsc{Abort}, \mathsf{sen}, \mathsf{sid})$ from S, send $(\textsc{Abort}, \mathsf{sid})$ to R and S. Ignore future messages with same sid.

5.4 Our Construction

We now present our construction of OT extension. We begin by briefly recalling the semi-honest OT construction of [LGdSG21], which is the starting point of our construction.

5.4.1 OT construction of [LGdSG21]

Let $(G, X, \star)$ be an EGA with x_0 being a publicly available element in the set X where reciprocal EGA assumption holds. Let $H : X \to \{0,1\}^\kappa$ be a hash function (modeled in the proof as a random oracle). Let $\mathcal{T} : X \to X$ denote the twist operation. Receiver R has input choice bit $b \in \{0,1\}$ and sender has inputs messages $(m_0, m_1) \in \{0,1\}^\kappa$. It is a tuple of five PPT algorithms $(\mathsf{Setup}, \mathsf{OTR}, \mathsf{OTS}_1, \mathsf{OTD})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample a trusted set element x_0 such that $\mathcal{T}(x_0) = x_0$. Sample $g \leftarrow_R G$ and output $\mathsf{crs} = x = g \star x_0$.
- $\mathsf{OTR}(\mathsf{crs}, \mathbf{b})$: Sample $r \leftarrow_R G$ and compute $z \in X$ as follows:

$$z = r \star x \text{ if } b = 0, \quad z = \mathcal{T}(r \star x) \text{ if } b = 1,$$

Output the receiver message $\mathsf{ot}_1 = z$ and the receiver state $\mathsf{st} = (b, r)$.

- $\mathsf{OTS}(\mathsf{crs}, \mathsf{ot}_1)$: Sample uniformly at random $s \leftarrow_R G$ and compute sender's OT message $y \in X$ and sender's random pads - $(a_0, a_1) \in \{0,1\}^\kappa$ as follows:

$$y = s \star x, \quad c_0 = H(s \star z) \oplus \mathsf{m}_0 \quad c_1 = H(s \star \mathcal{T}(z)) \oplus \mathsf{m}_1.$$

  Send the sender OT message as $\mathsf{ot}_2 = (y, c_0, c_1)$.

- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (b, r)$ and $\mathsf{ot}_2 = (y, c_0, c_1)$, and recover the output message $m_b = c_b \oplus H(r \star y)$.

Security against Malicious Sender. At a high level, the protocol is secure against a malicious sender since z perfectly hides b since $(r \star x)$ and $\mathcal{T}(r \star x)$ are statistically indistinguishable

$$\begin{aligned}
&H(\mathcal{T}(g \cdot r \star \mathcal{T}(g^{-1} \star y))) \oplus \gamma_1 \\
&\qquad = H(\mathcal{T}(g \cdot r \star \mathcal{T}(g^{-1} \star y))) \oplus H(s \star ((r \cdot g)^{-1} \star \mathcal{T}(x_0))) \oplus m_1 \\
&\qquad = H(\mathcal{T}(g \cdot r \star \mathcal{T}(g^{-1} \star y))) \oplus H(s \cdot r^{-1} \cdot g^{-1} \star x_0) \oplus m_1 \\
&\qquad = H((g \cdot r)^{-1} \star (g^{-1} \star y)) \oplus H(s \cdot r^{-1} \cdot g^{-1} \star x_0) \oplus m_1 \\
&\qquad = H((g \cdot r)^{-1} \star (s \star x_0)) \oplus H(s \cdot r^{-1} \cdot g^{-1} \star x_0) \oplus m_1 = m_1
\end{aligned}$$

A corrupt receiver cannot compute both m_0 and m_1 since it requires to query H on $(q_0, q_1) = (s \star z, s \star \mathcal{T}(z))$. Given q_1, one can compute $s^{-1} \star z = \mathcal{T}(z)$. This breaks the reciprocal EGA assumption since the adversary computes $(s \star z, s^{-1} \star z)$ where $y = s \star x$ is generated by the challenger after it receives adversarially generated set element $z \in X$. However, the simulator is unable to extract a corrupt receiver's input choice bit since it is statistically hidden.

Security against Malicious Receiver. To achieve security against a malicious receiver, the work of [LGdSG21] adds an interactive challenge-proof-verify mechanism. The sender computes a challenge that challenges the receiver to prove that it knows randomness r such that $z = r \star x$ or $z = \mathcal{T}(r \star x)$. Upon receiving the challenge, the receiver decrypts m_b and computes the proof using randomness r. It sends the proof to the sender, who verifies it and completes the protocol. The proof is sent in the third round of the protocol, thus blowing up the round complexity to three rounds. This approach successfully extracts a corrupt receiver's input if it computes a correct proof to the sender's challenger. However, their challenge-proof-verify mechanism incurs an additional overhead of 7 isogeny computation. We note that this 3 round maliciously secure OT construction suffices for simulation-based security but they would need an additional round for UC security. We refer to their Eurocrypt presentation for details.

5.4.2 Constructing OT Extension Protocols from Reciprocal (R)EGA

We build an inexpensive challenge-proof-verify mechanism on top of the above semi-honest by relying only on symmetric key operations to obtain custom OT protocols. These custom OT protocols are used to instantiate the maliciously secure base OT protocols in the [KOS15] (KOS) OT extension paradigm using ideas from [CSW20].

Observations from [CSW20]. The work of [CSW20] (abbreviated henceforth as CSW) made crucial observations that suffices for the base-OT protocols in KOS: 1) The base OT protocols are run in a batch of $\ell = \mathcal{O}(\kappa) > 3\mu$ OTs together. Simulation based security should hold for non-aborting parties for the batch together. 2) A corrupt sender is allowed to launch selective failure attack on the base-OTs since the receiver possesses random choice bits. 3) The base-OT protocols needs to satisfy simulation-based security only for non-aborting parties, in case of an abort semantic security suffices. The OT functionality $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ with selective failure attack, which is weaker than UC-OT functionality, suffices for the base OT in KOS We show a technique that builds upon the semi-honest OT protocol of [LGdSG21] to implement $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ against malicious adversaries. Our transformation only relies on cheap symmetric key operations. This reduces our isogeny computations for each base OT to 5 and it also yields the first OT extension protocol based on isogenies.

5.4.3 Overview of Our Construction

We build upon the semi-honest protocol of [LGdSG21]. Recall that their two round protocol (described in Sec. 5.4.1) is secure against a malicious sender and a semi-honest receiver since the simulator fails to extract the corrupt receiver's input. They add a challenge-proof-verify mechanism to tackle a malicious receiver but that doubles their isogeny computations. Instead, we take a different route and construct the same challenge-proof-verify mechanism by solely relying on symmetric key operations. Our mechanism is inspired from the the work of CSW and we describe it as follows.

Let us denote the two messages of the OT sender for the ith OT as $p_{0,i}$ and $p_{1,i}$ respectively. Let $H_1 : X \to \{0,1\}^\kappa, H_2 : \{0,1\}^\kappa \to \{0,1\}^\kappa, H_3 : \{0,1\}^{\ell\kappa} \to \{0,1\}^\kappa, H_4 : \{0,1\}^{2\kappa} \to \{0,1\}^\kappa$ be different hash functions (modeled in the proof as a random oracle). Let us denote the choice bit of the receiver for the ith OT as b_i. The sender constructs a challenge chall_i using the two messages as follows:

$$\mathsf{chall}_i = u_{0,i} \oplus u_{1,i}, \quad \text{where } u_{0,i} = H_2(i, p_{0,i}), \quad u_{1,i} = H_2(i, p_{1,i}).$$

The receiver is required to compute the response as $u_{0,i}$ and send it back to the sender as the proof. The receiver decrypts $p_{b_i,i}$ and computes $u_{0,i}$ as follows:

$$u_{0,i} = \mathsf{chall}_i \cdot b_i \oplus H_2(p_{b_i,i}).$$

Note that the receiver needs to query the random oracle H_2 in order to compute $u_{0,i}$ correctly and hence the simulator successfully extracts b_i if the receiver computes the correct response $u_{0,i}$. However, a corrupt sender can extract b_i by constructing chall_i maliciously. It samples a random chall'_i and sends it to the receiver. If the receiver responds with the correct $u_{0,i}$ then the sender sets $b_i = 0$ else it sets $b_i = 1$.

We tackle this problem by relying on the observation that the OT protocol can allow selective failure attack and it can allow the sender to guess $\mathcal{O}(\kappa)$ choice bits of the receiver. This suffices for the KOS base OT protocols. Using this observation we make the sender prove that the batch of ℓ challenges were correctly computed. The sender computes the response ans of receiver proof using a random oracle H_3 as follows:

$$\mathsf{ans} = H_3(u_{0,1}, u_{0,2}, \dots u_{0,\ell}).$$

The sender sends proof of correct computation by sending the proof $\mathsf{pf} = H_2(\mathsf{ans})$ to the receiver alongwith the challenger. The sender sets the output of ℓ random OTs as $(\mathbf{a}_0, \mathbf{a}_1)$ where $\mathbf{a}_0 =$

$\{a_{0,i}\}_{i\in[\ell]}$ and $\mathbf{a}_1 = \{a_{1,i}\}_{i\in[\ell]}$ is defined as follows for $i \in [\ell]$:

$$a_{0,i} = H_4(\mathsf{ans}, p_{0,i}), \quad a_{1,i} = H_4(\mathsf{ans}, p_{1,i}).$$

Upon receiving the sender's OT message, the receiver computes $p_{b_i,i}$ corresponding to its choice bit b_i. It computes $\{u_{0,i}\}_{i\in[\ell]}$ and recomputes ans to verify pf. If the verification succeeds then the receiver sends ans to the sender as the response and computes the OT output as $a_{b_i,i} = H_4(\mathsf{ans}, p_{b_i,i})$. If a corrupt receiver computes the correct ans then a simulator extracts every $\{b_i\}_{i\in[\ell]}$ by observing the queries made to H_2 and H_3. Without computing the correct ans the corrupt receiver cannot compute the OT output $a_{b_i,i}$. Hence, the simulator successfully extracts all the choice bits of the receiver if the receiver needs to compute the output of any single OT. Meanwhile, a corrupt sender can launch a selective failure attack only if it correctly guesses the value of receiver computed ans to verify pf. This is performed by guessing the $u_{0,i}$ values computed by the receiver and for that the sender needs to guess the receiver's choice bit in the OT protocols. The base OT protocols in KOS are random OTs. The sender guesses κ choice bits of the receiver with only $2^{-\kappa}$ probability. Thus, our OT protocol allows selective failure attack and it implements the $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality.

5.4.4 Details of Our Construction

Let $(G, X, \star)$ be an EGA with x_0 being a publicly available element in the set X where reciprocal EGA assumption holds. Also let $H_1 : X \to \{0,1\}^\kappa, H_2 : \{0,1\}^\kappa \to \{0,1\}^\kappa, H_3 : \{0,1\}^{\ell\kappa} \to \{0,1\}^\kappa, H_4 : \{0,1\}^{2\kappa} \to \{0,1\}^\kappa$ be different hash functions (modeled in the proof as a random oracle). Our construction is a tuple of five PPT algorithms $(\mathsf{Setup}, \mathsf{OTR}_0, \mathsf{OTS}_1, \mathsf{OTR}, \mathsf{OTS}_2)$:

- $\mathsf{Setup}(1^\lambda)$: Sample a trusted set element x_0 such that $\mathcal{T}(x_0) = x_0$. Sample $g \leftarrow_R G$ and output $\mathsf{crs} = x = g \star x_0$.

- $\mathsf{OTR}_1(\mathsf{crs}, \mathbf{b})$: Sample $\mathbf{r} \leftarrow_R G^\ell$ and compute $\mathbf{z} \in X^\ell$ as follows for $i \in [\ell]$:

  $$z_i = r_i \star x, \qquad \text{if } b_i = 0,$$

  $$z_i = \mathcal{T}(r_i \star x), \qquad \text{if } b_i = 1,$$

  Output the receiver message $\mathsf{ot}_1 = \mathbf{z}$ and the receiver state $\mathsf{st} = (\mathbf{b}, \mathbf{r})$.

- $\mathsf{OTS}_1(\mathsf{crs}, \mathsf{ot}_1)$: Sample uniformly at random $\mathbf{s} \leftarrow_R G^\ell$ and compute sender's OT message $\mathbf{y} \in X^\ell$ and sender's random inputs messages as $(\mathbf{p}_0, \mathbf{p}_1) \in \{0,1\}^{\kappa\times\ell}$ as follows for $i \in [\ell]$:

  $$y_i = s_i \star x, \quad p_{0,i} = H_1(i, s_i \star z_i) \quad p_{1,i} = H_1(i, s_i \star \mathcal{T}(z_i)).$$

  Compute the challenge **chall** for receiver proof as follows for $i \in [\ell]$:

  $$\mathsf{chall}_i = u_{0,i} \oplus u_{1,i}, \quad \text{where } u_{0,i} = H_2(i, p_{0,i}), \quad u_{1,i} = H_2(i, p_{1,i}).$$

  Compute the response ans of receiver proof as follows:

  $$\mathsf{ans} = H_3(u_{0,1}, u_{0,2}, \ldots u_{0,\ell}).$$

  Compute the sender's proof $\mathsf{pf} = H_2(\mathsf{ans})$. Send the sender OT message as $\mathsf{ot}_2 = (\mathbf{y}, \mathbf{chall}, \mathsf{pf})$. Store $(\mathsf{ans}, \mathbf{p}_0, \mathbf{p}_1)$ as the internal state.

- $\mathsf{OTR}_2(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (\mathbf{b}, \mathbf{r})$ and $\mathsf{ot}_2 = (\mathbf{y}, \mathbf{chall}, \mathsf{pf})$, and recover the output pads $\mathbf{p} = \{p_i\}_{i \in [\ell]}$ as follows for $i \in [\ell]$:
$$p_i = H_1(r_i \star y_i).$$
Compute the intermediate proof response as follows for $i \in [\ell]$:
$$u'_i = \mathsf{chall}_i \cdot b_i \oplus H_2(i, p_i)$$
Compute the receiver's proof response ans' as follows:
$$\mathsf{ans}' = H_3(u'_1, u'_2, \ldots u'_\ell)$$
The receiver aborts if $H_2(\mathsf{ans}') \neq \mathsf{pf}$. Else, the receiver responds to the sender's challenge by sending $\mathsf{ot}_3 = \mathsf{ans}'$ to the sender. The receiver computes the OT output as $\mathbf{m} = \{m_i\}_{i \in [\ell]}$ for $i \in [\ell]$:
$$m_i = H_4(\mathsf{ans}', p_i)$$
Output $(\mathbf{b}, \mathbf{m})$ as the random OT receiver output.

- $\mathsf{OTS}_2(\mathsf{ans}, \mathsf{ot}_3)$: Parse $\mathsf{ot}_3 = \mathsf{ans}'$. The sender aborts if $\mathsf{ans}' \neq \mathsf{ans}$. Else, the sender sets the output as $(\mathbf{a}_0, \mathbf{a}_1)$ where $\mathbf{a}_0 = \{a_{0,i}\}_{i \in [\ell]}$ and $\mathbf{a}_1 = \{a_{1,i}\}_{i \in [\ell]}$ is defined as follows for $i \in [\ell]$:
$$a_{0,i} = H_4(\mathsf{ans}, p_{0,i}), \quad a_{1,i} = H_4(\mathsf{ans}, p_{1,i}).$$

Correctness. Correctness of the scheme follows by inspection. It can be verified that $m_i = a_{b_i,i}$ for $i \in [\ell]$ for honest execution.

Security Proof. The security of our protocol is summarized by the following theorem.

Theorem 9. *Given a set element $x \in X$ and a group G, let $G \star X$ be the distribution on X of $g \star x$ for $g \leftarrow_R G$, and let $\mathcal{T}(G \star X)$ be the distribution on x of $\mathcal{T}(g \star x)$ for $g \leftarrow_R G$. If $g \star x$ and $\mathcal{T}(g \star x)$ are statistically indistinguishable and reciprocal EGA assumption holds in $(G, X, \star)$ then the above protocol securely implements $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality (Figure. 4) against a maliciously corrupt sender and a maliciously corrupt non-aborting receiver in the random oracle model.*

We prove security of our protocol by considering two corruption cases.

Security against Malicious Sender. We prove simulation based security against a maliciously corrupt sender by proving Lemma. 3.

Lemma 3. *Given a set element $x \in X$ and a group G, let $G \star X$ be the distribution on X of $g \star x$ for $g \leftarrow_R G$, and let $\mathcal{T}(G \star X)$ be the distribution on x of $\mathcal{T}(g \star x)$ for $g \leftarrow_R G$. If $g \star x$ and $\mathcal{T}(g \star x)$ are statistically indistinguishable then the above protocol securely implements $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality (Figure. 4) against a maliciously corrupt sender in the random oracle model.*

Proof. Our simulator against a corrupt sender is as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g \leftarrow_R G$ and output $\mathsf{crs} = x = g \star x_0$. The simulator possesses the trapdoor g.

- $\mathsf{OTR}_1(\mathsf{crs}, \mathbf{b})$: Perform this step following honest receiver algorithm. Sample uniformly at random $\mathbf{b} \leftarrow_R \{0,1\}^\ell$ and $\mathbf{r} \leftarrow_R G^\ell$ and compute $\mathbf{z} \in X^\ell$ as follows for $i \in [\ell]$:

$$z_i = r_i \star x, \quad \text{if } b_i = 0,$$

$$z_i = \mathcal{T}(r_i \star x), \quad \text{if } b_i = 1,$$

  Output the receiver message $\mathsf{ot}_1 = \mathbf{z}$ and the receiver state $\mathsf{st} = (\mathbf{b}, \mathbf{r})$.

- $\mathsf{OTS}_1(\mathsf{crs}, \mathsf{ot}_1)$: The malicious sender sends OT message as $\mathsf{ot}_2 = (\mathbf{y}, \mathbf{chall}, \mathsf{ans})$.

- $\mathsf{OTR}_2(\mathsf{st}, \mathsf{ot}_2)$: Parse $\mathsf{st} = (\mathbf{b}, \mathbf{r})$ and $\mathsf{ot}_2 = (\mathbf{y}, \mathbf{chall}, \mathsf{pf})$, and recover the output messages $\mathbf{m}$ as follows:

$$m_i = H_1(r_i \star y_i) = p_{b_i,i}.$$

  Compute the receiver proof response as follows :

$$u'_i = \mathsf{chall}_i \cdot b_i \oplus H_2(i, m_i) \quad (\text{ for } i \in [\ell])$$

$$\mathsf{ans}' = H_3(u'_1, u'_2, \ldots u'_\ell)$$

  The simulator aborts if $H_2(\mathsf{ans}') \neq \mathsf{pf}$. Else, the receiver responds to the sender's challenge by sending $\mathsf{ot}_3 = \mathsf{ans}'$ to the sender. Extract $p_{\overline{b_i},i}$ as follows:

$$p_{\overline{b_i},i} = H_1(\mathcal{T}(g \cdot r_i \star \mathcal{T}(g^{-1} \star y_i))).$$

  The simulator observes the random oracle queries made by the sender to H_1, H_2 and H_3, and performs the following:

  - For $i \in [\ell]$, if the sender has queried $(i, \mathbf{p}_{d_i,i})$ and not queried $(i, \mathbf{p}_{\overline{d_i},i})$ to H_2, for $d_i \in \{0,1\}$, then invoke $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ with a guess for selective failure attack as - $(\textsc{Guess}, \mathsf{sen}, i, d_i)$. If it returns $(\textsc{Cheat-Detected}, \mathsf{S})$ then abort else continue. The simulator aborts if the malicious sender guesses $> \mu$ choice bits successfully in the protocol.
  - For $i \in [\ell]$, if S has not queried both $(i, p_{0,i})$ and $(i, p_{1,i})$ to H_2 then invoke $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ with - $(\textsc{Guess}, \mathsf{sen}, i, \perp)$ and abort the protocol.
  - Aborts if S has not queried $(u'_1, u'_2, \ldots, u'_\ell)$ to H_3.
  - Aborts if S has not queried ans' to H_2.

  If all the checks pass then the simulator computes $\mathbf{a}_0 = \{a_{0,i}\}_{i\in[\ell]}$ and $\mathbf{a}_1 = \{a_{1,i}\}_{i\in[\ell]}$ as follows for $i \in [\ell]$:

$$a_{0,i} = H_4(\mathsf{ans}', p_{0,i}), \quad a_{1,i} = H_4(\mathsf{ans}', p_{1,i}).$$

  The simulator invokes $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ with input $(\textsc{Transfer}, \mathsf{S}, \mathsf{sid}, (\mathbf{a}_0, \mathbf{a}_1))$ and completes the simulation.

- $\mathsf{OTS}_2(*, \mathsf{ot}_3)$: Performs its own adversarial algorithm.

The corrupt sender forwards its view to the simulator which is the ideal world view of the adversary. We argue security against a maliciously corrupted sender by showing that the real world view of the adversary is indistinguishable from the ideal world view of the adversary as follows.

- $\texttt{Hyb}_0$: Real world execution of the protocol.

- $\texttt{Hyb}_1$: Same as $\texttt{Hyb}_0$, except the simulator aborts if the malicious sender has not queried ans' to H_2. The adversarial sender distinguishes the two hybrids if it predicts $\mathsf{pf} = H_2(\mathsf{ans}')$ without querying ans' to H_2. This event occurs with negligible probability in the random oracle model.

- $\texttt{Hyb}_2$: Same as $\texttt{Hyb}_1$, except the simulator aborts if the malicious sender has not queried $\mathbf{u}' = (u'_1, u'_2, \ldots, u'_\ell)$ to H_3. The adversarial sender distinguishes the two hybrids if it predicts $\mathsf{ans}' = H_3(\mathbf{u}')$ without querying $\mathbf{u}'$ to H_3 or it finds a collision - $(\mathbf{u}', \mathbf{u}'')$, in H_3 such that $\mathsf{ans}' = H_3(\mathbf{u}') = H_2(\mathbf{u}'')$ and $\mathbf{u}' \neq \mathbf{u}''$. Both events occur with negligible probability in the random oracle model.

- $\texttt{Hyb}_3$: Same as $\texttt{Hyb}_2$, except the simulator invokes the $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality with (GUESS, sen, i,$\perp$) and aborts the protocol if the sender has not queried both $(i, p_{0,i})$ and $(i, p_{1,i})$ to H_2 for some $i \in [\ell]$. The adversarial sender distinguishes the two hybrids if it predicts $H_2(i, p_{0,i})$ or $H_2(i, p_{1,i})$ without querying H_2 on $(i, p_{0,i})$ or $(i, p_{1,i})$ respectively. This event occurs with negligible probability in the random oracle model.

- $\texttt{Hyb}_4$: Same as $\texttt{Hyb}_3$, except the simulator extracts both $(\mathbf{a}_0, \mathbf{a}_1)$ following the simulation algorithm and simulates the selective failure attacks as per the simulation algorithm. The simulator aborts if the adversarial sender launches selective failure attack on $> \mu$ OTs. The receiver OT message ot_1 statistically hides $\mathbf{b}$ since $r_i \star x$ and $\mathcal{T}(r_i \star x)$ are statistically indistinguishable. The only other way a corrupt sender distinguishes between the two is when it passes all the checks in real protocol, whereas in the ideal world the simulator aborts. This occurs with statistically negligible probability $2^{-\mu}$ since the adversary has to successfully launch a selective failure attack on $> \mu$ invocations of $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$. This is the ideal world execution of the protocol and it completes our simulation.

This completes the proof of Lemma 3. $\square$

Security against a Malicious Receiver. Next, we show that our protocol provides simulation based security against a non-aborting malicious receiver by proving Lemma. 4.

Lemma 4. *If the reciprocal EGA assumption holds in* $(G, X, \star)$ *then the above protocol securely implements* $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ *functionality (Figure. 4) against a maliciously corrupt non-aborting receiver in the random oracle model.*

Proof. Our simulation algorithm against a corrupt receiver is as follows:

- $\mathsf{Setup}(1^\lambda)$: Sample $g \leftarrow_R G$ and output $\mathsf{crs} = x = g \star x_0$.

- $\mathsf{OTR}_1(\mathsf{crs}, \mathbf{b})$: The malicious receiver sends $\mathsf{ot}_1 = \mathbf{z}$.

- $\mathsf{OTS}_1(\mathsf{crs}, \mathsf{ot}_1)$: Sample uniformly at random $\mathbf{s} \leftarrow_R G^\ell$ and compute sender's OT message $\mathbf{y} \in X^\ell$ and sender's random pads as $(\mathbf{p}_0, \mathbf{p}_1) \in \{0,1\}^{\kappa \times \ell}$ as follows for $i \in [\ell]$:

$$y_i = s_i \star x, \quad p_{0,i} = H_1(i, s_i \star z_i) \quad p_{1,i} = H_1(i, s_i \star \mathcal{T}(z_i)).$$

Compute the challenge **chall** for receiver proof as follows for $i \in [\ell]$:

$$\mathsf{chall}_i = u_{0,i} \oplus u_{1,1}, \quad \text{where } u_{0,i} = H_2(i, p_{0,i}), \quad u_{1,i} = H_2(i, p_{1,i}).$$

Compute the response ans of receiver proof as follows:

$$\mathsf{ans} = H_3(u_{0,1}, u_{0,2}, \ldots u_{0,\ell}).$$

Compute the sender's proof $\mathsf{pf} = H_2(\mathsf{ans})$. Send the sender OT message as $\mathsf{ot}_2 = (\mathbf{y}, \mathbf{chall}, \mathsf{ans})$. Store ans as the internal state.

- $\mathsf{OTR}_2(\mathsf{st}, \mathsf{ot}_2)$: The corrupt receiver sends $\mathsf{ot}_3 = \mathsf{ans}'$.

- $\mathsf{OTS}_2(\mathsf{ans}, \mathsf{ot}_3)$: Parse $\mathsf{ot}_3 = \mathsf{ans}'$ and abort if $\mathsf{ans} \neq \mathsf{ans}'$. The simulator extracts the receiver's input $\mathbf{b}$ as follows for $i \in [\ell]$:

  – Set $b_i = 0$ if receiver has queried $(i, s_i \star z_i)$ to H_1.

  – Set $b_i = 1$ if receiver has queried $(i, s_i \star \mathcal{T}(z_i))$ to H_1.

  The simulator aborts if one of the following event occurs:

  – For $i \in [\ell]$, the receiver did not query both $(i, s_i \star z_i)$ and $(i, s_i \star \mathcal{T}(z_i))$ to H_1.

  – For $i \in [\ell]$, the receiver queried both $(i, s_i \star z_i)$ and $(i, s_i \star \mathcal{T}(z_i))$ to H_1.

  – For any $i \in [\ell]$, the receiver did not query both $p_{0,i}$ and $p_{1,i}$ to H_2.

  – The receiver did not query $(u_{0,1}, u_{0,2}, \ldots, u_{0,\ell})$ to H_3.

  Invoke $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ with input $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid}, \mathbf{b})$ as a corrupt receiver to obtain $\mathbf{m} = \{m_i\}$ as output. The simulator programs the random oracle H_4 as follows such that the receiver obtains the correct output:

$$H_4(\mathsf{ans}, p_{b_i,i}) = m_i \text{ (for } i \in [\ell]).$$

- $\mathtt{Hyb}_0$: Real world execution of the protocol.

- $\mathtt{Hyb}_1$: Same as $\mathtt{Hyb}_0$, except the simulator aborts if the receiver did not query $(u_{0,1}, \ldots, u_{0,\ell})$ to H_3. The receiver distinguishes between the two hybrids if it successfully predicts the output of $H_3(u_{0,1}, u_{0,2}, \ldots, u_{0,\ell})$ as ans without querying. This event occurs with negligible probability in the random oracle model.

- $\mathtt{Hyb}_2$: Same as $\mathtt{Hyb}_1$, except the simulator aborts if for any $i \in [\ell]$, the receiver did not query both $p_{0,i}$ and $p_{1,i}$ to H_2. The receiver distinguishes between the two hybrids if it successfully predicted the output of H_2 without querying. This event occurs with negligible probability in the random oracle model.

- $\mathtt{Hyb}_3$: Same as $\mathtt{Hyb}_2$, except the simulator aborts if for any $i \in [\ell]$, the receiver did not query both $(i, s_i \star z_i)$ and $(i, \mathcal{T}(s_i \star z_i))$ to H_1. The receiver distinguishes between the two hybrids if it successfully predicted the output of H_1 without querying. This event occurs with negligible probability in the random oracle model.

- $\texttt{Hyb}_4$: Same as $\texttt{Hyb}_3$, except the simulator aborts if for any $i \in [\ell]$, the receiver queried both $(i, s_i \star z_i)$ and $(i, \mathcal{T}(s_i \star z_i))$ to H_1. Else, the simulator uniquely extracts the receiver's choice bits and invokes $\mathcal{F}_{\mathsf{SF\text{-}ROT}}$ functionality with receiver's input to obtain the output and it programs H_4 such that receiver gets the correct output. If the receiver successfully distinguishes between the two hybrids then one can break the reciprocal CSIDH assumption as follows. When the receiver sends z_i, forward $z^* = z_i$ to the reciprocal CSIDH challenger and receive $y^* = s^* \star x$ as the challenge and return $y_i = y^*$ to the receiver as part of the sender OT message. When receiver queries both $(i, s_i \star z_i)$ and $(i, s_i \star \mathcal{T}(z_i))$ to H_1, return $(s^* \star z^*) = (s_i \star z_i)$ and $(s^{*-1} \star z^*) = \mathcal{T}(s_i \star \mathcal{T}(z_i))$ to the reciprocal CSIDH challenger as the response. This breaks reciprocal CSIDH assumption and hence proves that the two hybrids are indistinguishable. This is our ideal execution of the protocol and it completes our security proof.

This completes the proof of Lemma 4, and hence, the proof of Theorem 9. □

Further Optimizations. It can be observed that the sender can reuse the randomness s for multiple OT protocols by using reusing the same y for all the OT protocols. This translates into a $\mathsf{poly}(\kappa)$ loss in the security parameter since the reduction to reciprocal EGA assumption needs to guess the session where a corrupt receiver breaks the assumption. The security loss can be compensated by increasing the security parameter accordingly. This optimization reduces the number of isogeny computations to 4 for each OT. Meanwhile, the semi-honest OT protocol of [LGdSG21] requires 5 isogeny computations.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[ADDS21] Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In *PKC 2021*, volume 12711, pages 261–289, 2021.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020, Part II*, LNCS, pages 411–439. Springer, Heidelberg, December 2020.

[AEK$^+$22] Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In *CRYPTO 2022*, volume 13508 of *Lecture Notes in Computer Science*, pages 699–728. Springer, 2022.

[AMPS21] Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Pratik Sarkar. Two-round adaptively secure MPC from isogenies, lpn, or CDH. In *ASIACRYPT 2021*, volume 13091 of *Lecture Notes in Computer Science*, pages 305–334. Springer, 2021.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 370–390. Springer, Heidelberg, November 2018.

[BDK$^+$20] Niklas Büscher, Daniel Demmler, Nikolaos P. Karvelas, Stefan Katzenbeisser, Juliane Krämer, Deevashwer Rathee, Thomas Schneider, and Patrick Struck. Secure two-party computation in a quantum world. In *ACNS 2020*, pages 461–480, 2020.

[BDK$^+$21] Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. *IACR Cryptol. ePrint Arch.*, page 1366, 2021. (Accepted in Eurocrypt 2022).

[BF22] Nir Bitansky and Sapir Freizeit. Statistically sender-private OT from LPN and derandomization. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022*, volume 13508 of *Lecture Notes in Computer Science*, pages 699–728. Springer, 2022.

[BGJ$^+$18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 459–487. Springer, Heidelberg, August 2018.

[BKM$^+$21] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Antonio Sanso. Cryptanalysis of an oblivious PRF from supersingular isogenies. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 160–184. Springer, 2021.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019, Part I*, LNCS, pages 227–247. Springer, Heidelberg, December 2019.

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In *ASIACRYPT 2020, Part II*, LNCS, pages 520–550. Springer, Heidelberg, December 2020.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018.

[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.

[BPS22] Saikrishna Badrinarayanan, Sikhar Patranabis, and Pratik Sarkar. Statistical security in two-party computation revisited. *IACR Cryptol. ePrint Arch. (To appear in TCC 2022)*, page 1190, 2022.

[BY91] Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Heidelberg, August 1991.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *TCC 2020, Part II*, LNCS, pages 291–319. Springer, Heidelberg, March 2020.

[CD22] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). *IACR Cryptol. ePrint Arch.*, page 975, 2022.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

[CLM+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, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[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 Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS, pages 523–548. Springer, Heidelberg, May 2020.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003.

[CSV20] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional Diffie-Hellman problem for class group actions using genus theory. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2020, Part II*, LNCS, pages 92–120. Springer, Heidelberg, August 2020.

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Blazing fast OT for three-round UC OT extension. In *PKC 2020, Part II*, LNCS, pages 299–327. Springer, Heidelberg, 2020.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

[DGH+20] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS, pages 768–797. Springer, Heidelberg, May 2020.

[DMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *ASIACRYPT 2019, Part I*, LNCS, pages 248–277. Springer, Heidelberg, December 2019.

[DNM12] Bernardo Machado David, Anderson C. A. Nascimento, and Jörn Müller-Quade. Universally composable oblivious transfer from lossy encryption and the McEliece assumptions. In Adam Smith, editor, *ICITS 12*, volume 7412 of *LNCS*, pages 80–99. Springer, Heidelberg, August 2012.

[dSGOPS20] Cyprien Delpech 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, *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*, volume 12579 of *Lecture Notes in Computer Science*, pages 235–258. Springer, 2020.

[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*, volume 5155 of *LNCS*, pages 107–117. Springer, Heidelberg, August 2008.

[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, *CRYPTO'82*, pages 205–210. Plenum Press, New York, USA, 1982.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[FMV19] Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In *TCC 2019, Part I*, LNCS, pages 111–130. Springer, Heidelberg, March 2019.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Heidelberg, March 2009.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[KM20] Dakshita Khurana and Muhammad Haris Mughees. On statistical security in two-party computation. In *TCC 2020, Part II*, LNCS, pages 532–561. Springer, Heidelberg, March 2020.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

[LGdSG21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, efficient and uc-secure isogeny-based oblivious transfer. In *EUROCRYPT 2021*, pages 213–241, 2021.

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 353–365. Springer, Heidelberg, August 1991.

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116, 1978.

[MM22] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. *IACR Cryptol. ePrint Arch.*, page 1026, 2022.

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In *ACM CCS 2019*, pages 309–326. ACM Press, 2019.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003.

[Pet17] Christophe Petit. Faster algorithms for isogeny problems using torsion point images. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 330–353. Springer, Heidelberg, December 2017.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd FOCS*, pages 366–375. IEEE Computer Society Press, November 2002.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[Qua20] Willy Quach. UC-secure OT from LWE, revisited. In *SCN 20*, LNCS, pages 192–211. Springer, Heidelberg, 2020.

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. https://eprint.iacr.org/2005/187.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[Vit18] Vanessa Vitse. Simple oblivious transfer protocols compatible with kummer and supersingular isogenies. Cryptology ePrint Archive, Report 2018/709, 2018. https://eprint.iacr.org/2018/709.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.