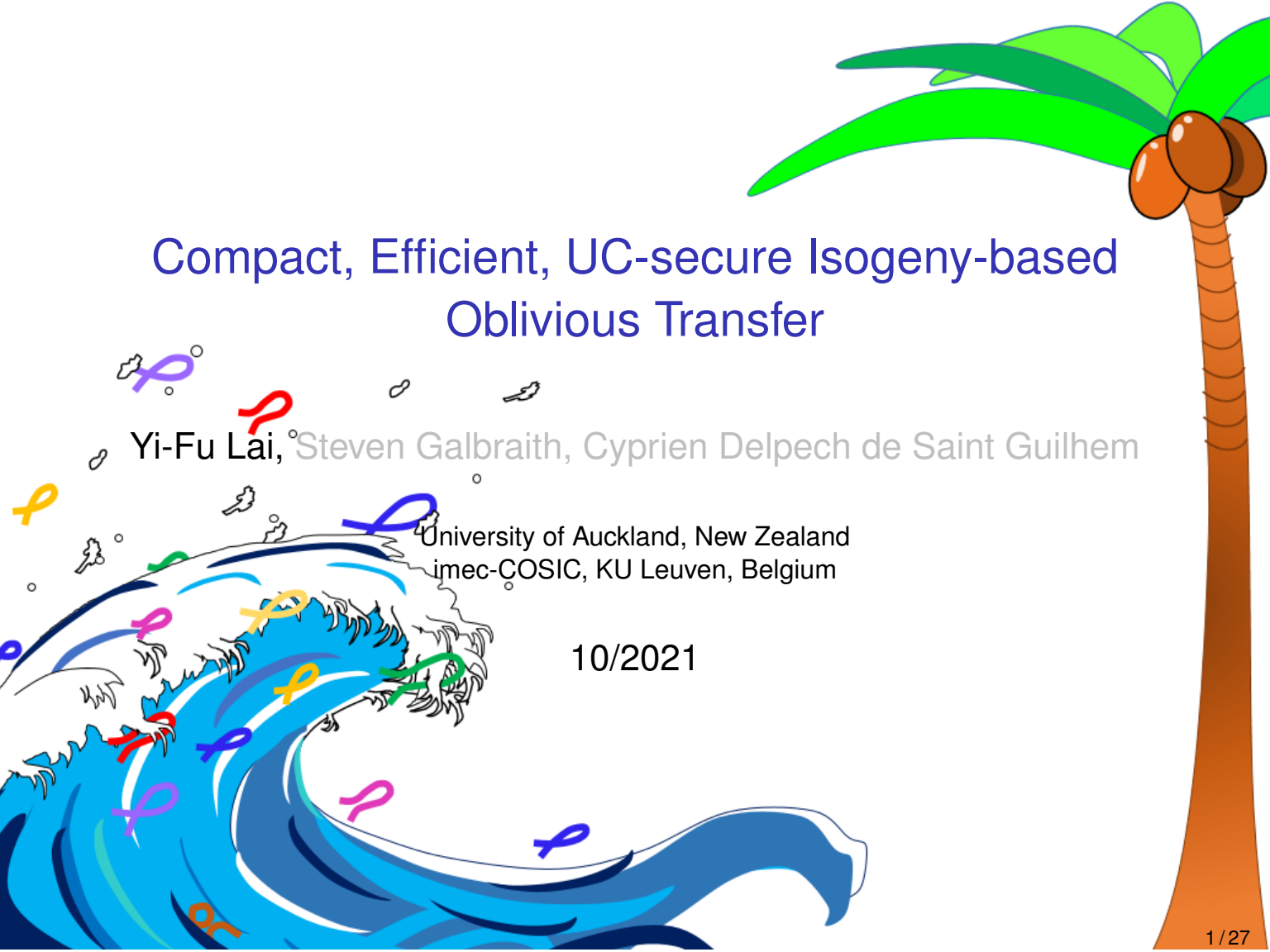
Compact, Efficient, UC-secure Isogeny-based Oblivious Transfer

Yi-Fu Lai, Steven Galbraith, Cyprien Delpech de Saint Guilhem

University of Auckland, New Zealand
imec-COSIC, KU Leuven, Belgium

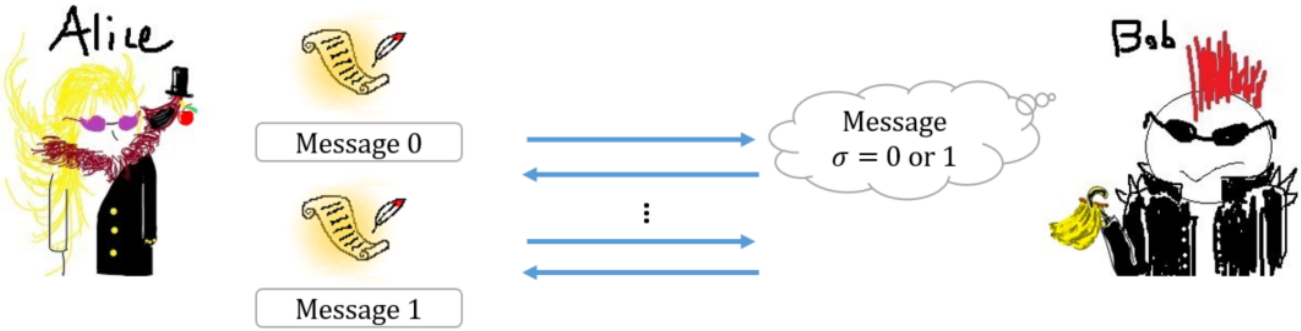
10/2021



Content

Content

Oblivious Transfer



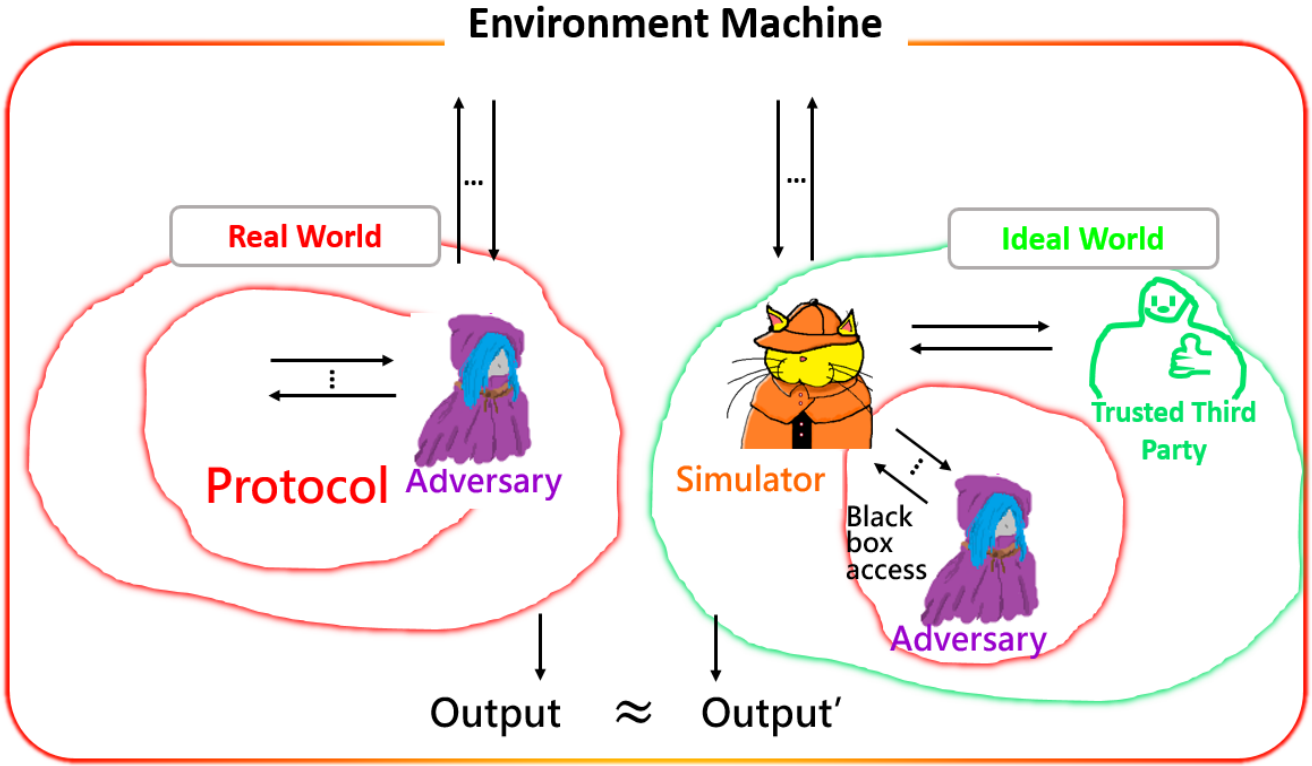
The “classical” requirements for an oblivious transfer scheme are:

1. Bob gets one and only one message.
2. Alice doesn’t know Bob’s choice.

Given the importance of OT as a cryptographic tool, we need a stronger notion taking composition into account.

Universally-Composible (UC) Security

UC security proposed by Canetti [Can01] is a simulation based security notion. The simulator doesn't simulate for the adversary but for the environment machine.



Research Question

- Two isogeny based previous works with acceptable security are [dSGOPS18] and [Vit19]. They are UC-secure against semi-honest adversaries.
- By adding an UC-secure ZK proof protocol or using the generic transformation like [DGH+20], the protocol can upgrade to UC-secure against malicious adversaries while it will take $poly(\lambda)$ isogeny computations.

Can we have an isogeny-based OT UC-secure against malicious adversaries taking a constant number of isogeny computations?

Content

CSIDH

Given an odd prime p. A Montgomery curve over $\mathbb{F}_p$ is defined as

$$E_A : y^2 = x^3 + Ax^2 + x, A \in \mathbb{F}_p.$$

- An elliptic curve E_A defined over $\mathbb{F}_p$ is said to be supersingular if $|E_A(\mathbb{F}_p)| = p + 1$.

CSIDH

Given an odd prime p. A Montgomery curve over $\mathbb{F}_p$ is defined as

$$E_A : y^2 = x^3 + Ax^2 + x, A \in \mathbb{F}_p.$$

- An elliptic curve E_A defined over $\mathbb{F}_p$ is said to be supersingular if $|E_A(\mathbb{F}_p)| = p + 1$.

Let $\mathcal{O}$ be an order of an imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ and $\pi \in \mathcal{O}$.

CSIDH

Given an odd prime p. A Montgomery curve over $\mathbb{F}_p$ is defined as

$$E_A : y^2 = x^3 + Ax^2 + x, A \in \mathbb{F}_p.$$

- An elliptic curve E_A defined over $\mathbb{F}_p$ is said to be supersingular if $|E_A(\mathbb{F}_p)| = p + 1$.

Let $\mathcal{O}$ be an order of an imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ and $\pi \in \mathcal{O}$. Let $\mathcal{E}_p(\mathcal{O}, \pi)$ collect all (supersingular) curves over $\mathbb{F}_p$, modulo $\mathbb{F}_p$-isomorphism, whose endomorphism rings over $\mathbb{F}_p$ are isomorphic to $\mathcal{O}$ where π corresponds to the Frobenius map.

CSIDH

Given an odd prime p. A Montgomery curve over $\mathbb{F}_p$ is defined as

$$E_A : \ y^2 = x^3 + Ax^2 + x, A \in \mathbb{F}_p.$$

- An elliptic curve E_A defined over $\mathbb{F}_p$ is said to be supersingular if $|E_A(\mathbb{F}_p)| = p + 1$.

Let $\mathcal{O}$ be an order of an imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ and $\pi \in \mathcal{O}$. Let $\mathcal{E}_p(\mathcal{O}, \pi)$ collect all (supersingular) curves over $\mathbb{F}_p$, modulo $\mathbb{F}_p$-isomorphism, whose endomorphism rings over $\mathbb{F}_p$ are isomorphic to $\mathcal{O}$ where π corresponds to the Frobenius map.

- We know that the ideal class group $Cl(\mathcal{O})$ acts freely and transitively on $\mathcal{E}_p(\mathcal{O}, \pi)$.

Quadratic Twist

The quadratic twist of $E_A/\mathbb{F}_p$ is defined as

$$E_A^t : \; dy^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p,$$

where $d^{(\frac{p-1}{2})} = -1 \mod p$.

Quadratic Twist

The quadratic twist of $E_A/\mathbb{F}_p$ is defined as

$$E_A^t : \ dy^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p,$$

where $d^{(\frac{p-1}{2})} = -1 \mod p$.

- We have $(a * E)^t = a^{-1} * E^t$.
- Moreover, when $p = 3 \mod 4$, we have $E_0^t = E_0$ and $E_A^t = E_{-A}$ ($\mathbb{F}_p$-isomorphic).
- Accordingly, when $p = 3 \mod 4$, we have

$$(a * E_0)^t = a^{-1} * E_0 \text{ for any } a \in Cl.$$

Assumptions

We simplify $Cl(\mathcal{O})$ as Cl, $\mathcal{E}_p(\mathcal{O}, \pi)$ as $\mathcal{E}$ and write $a \in Cl$, $E \in \mathcal{E}$ (ignoring the class notation). Assume we have uniform sampling over Cl, denoted by $\leftarrow Cl$.

Computational CSIDH problem

Fixed $E \in \mathcal{E}$. Given $(a * E, b * E)$ where $a, b \leftarrow Cl$. Find $ab * E$.

Square CSIDH problem

Fixed $E \in \mathcal{E}$. Given $(a * E)$ where $a \leftarrow Cl$. Find $a^2 * E$

Assumptions

We simplify $Cl(\mathcal{O})$ as Cl, $\mathcal{E}_p(\mathcal{O}, \pi)$ as $\mathcal{E}$ and write $a \in Cl$, $E \in \mathcal{E}$ (ignoring the class notation). Assume we have uniform sampling over Cl, denoted by $\leftarrow Cl$.

Computational CSIDH problem

Fixed $E \in \mathcal{E}$. Given $(a * E, b * E)$ where $a, b \leftarrow Cl$. Find $ab * E$.

Square CSIDH problem

Fixed $E \in \mathcal{E}$. Given $(a * E)$ where $a \leftarrow Cl$. Find $a^2 * E$

Given the order of the group Cl, they are equivalent:

- A proof for the case p=3 mod 4 can be found in [Fel19].
- A full proof for the case p=1 mod 4 can be found in our appendix.

Reciprocal CSIDH Problem

The reciprocal CSIDH problem is a 2-round experiment as follows:

$a \leftarrow Cl$

Public Curve: E

X

$A = a * E$

Supersingular curve $X \in \mathcal{E}$

- Output: $(a * X, a^{-1} * X)$

Equivalence-I

The reciprocal CSIDH Problem is as hard as the square CSIDH problem.

Equivalence-I

The reciprocal CSIDH Problem is as hard as the square CSIDH problem.

Reciprocal CSIDH Problem $\leq$ Square CSIDH problem:

1. Say the public curve is $E \in \mathcal{E}$, commit to $X = E$ and obtain a challenge $a * E$. (Then the task is to find $(a * E, a^{-1} * E)$.)

Equivalence-I

The reciprocal CSIDH Problem is as hard as the square CSIDH problem.

Reciprocal CSIDH Problem $\leq$ Square CSIDH problem:

1. Say the public curve is $E \in \mathcal{E}$, commit to $X = E$ and obtain a challenge $a * E$. (Then the task is to find $(a * E, a^{-1} * E)$.)
2. Invoke the oracle with $(a * E, E)$, obtain $C \in \mathcal{E}$ and output $(a * E, C)$.

Equivalence-I

The reciprocal CSIDH Problem is as hard as the square CSIDH problem.

Reciprocal CSIDH Problem $\leq$ Square CSIDH problem:

1. Say the public curve is $E \in \mathcal{E}$, commit to $X = E$ and obtain a challenge $a * E$. (Then the task is to find $(a * E, a^{-1} * E)$.)
2. Invoke the oracle with $(a * E, E)$, obtain $C \in \mathcal{E}$ and output $(a * E, C)$.

Correctness: Write $E = a^{-1} * (a * E)$, we have

$$C = a^{-2} * (a * E) = a^{-1} * E. \quad \square$$

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

1. Say the challenge is $(E, a * E)$ where $E \in \mathcal{E}$. (Then the task is to find $a^2 * E$.)

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

1. Say the challenge is $(E, a * E)$ where $E \in \mathcal{E}$. (Then the task is to find $a^2 * E$.)
2. Invoke the oracle with the public curve $a * E$. After the oracle commits to $X \in \mathcal{E}$, give the challenge E to the oracle.

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

1. Say the challenge is $(E, a * E)$ where $E \in \mathcal{E}$. (Then the task is to find $a^2 * E$.)
2. Invoke the oracle with the public curve $a * E$. After the oracle commits to $X \in \mathcal{E}$, give the challenge E to the oracle.
3. Obtain (X_1, X_2) from the oracle, rewind the oracle, and replace the challenge to be X_1.

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

1. Say the challenge is $(E, a * E)$ where $E \in \mathcal{E}$. (Then the task is to find $a^2 * E$.)
2. Invoke the oracle with the public curve $a * E$. After the oracle commits to $X \in \mathcal{E}$, give the challenge E to the oracle.
3. Obtain (X_1, X_2) from the oracle, rewind the oracle, and replace the challenge to be X_1.
4. Obtain (X_1', X_2') from the oracle. Output X_2'.

Equivalence-II

Reciprocal CSIDH Problem $\geq$ Square CSIDH problem: We can show this by using the *rewinding argument*.

1. Say the challenge is $(E, a * E)$ where $E \in \mathcal{E}$. (Then the task is to find $a^2 * E$.)
2. Invoke the oracle with the public curve $a * E$. After the oracle commits to $X \in \mathcal{E}$, give the challenge E to the oracle.
3. Obtain (X_1, X_2) from the oracle, rewind the oracle, and replace the challenge to be X_1.
4. Obtain (X_1', X_2') from the oracle. Output X_2'.

Correctness: Since $E = a^{-1} * (a * E)$, we have $X_1 = a^{-1} * X$. Thanks to transitive action, we can write $X = x * E$, also write $X_1 = (a^{-2}x) * (a * E)$. Hence,

$$X_2' = (a^2 x^{-1}) * X = a^2 * E. \quad \square$$

Content

3-Round Construction

Enc: an IND-CPA symmetric key encryption scheme.

Public Curve: E

$A = a * E$

B

$Enc_{k_0}(m_0)$

$Enc_{k_1}(m_1)$

Input: (m_0, m_1)

$a \xleftarrow{\$} Cl$

$k_0 = a * B$

$k_1 = a^{-1} * B$

Input: $\sigma \in \{0,1\}$

$b \xleftarrow{\$} Cl$

If $\sigma = 0$, set

$B = b * E$,

o.w. set

$B = b * A$.

3-Round Construction -Idea

Intuitively, the idea is as like (not a rigorous proof):

$a * E$

E

$a^2b * E$

$ab * E$

$b * E$

$a^{-1}b * E$

Decryption keys for $\sigma = 1$

To get $a^2b * E$
=To get $a^2 * E$.

Decryption keys for $\sigma = 0$

Quadratic Twist Round-Compression

Enc: IND-CPA symmetric key encryption scheme.
H: hash fuction

Public Curve: E

B ←

Input: (m_0, m_1)

$a \leftarrow Cl$

$k_0 = a * B$

$k_1 = a^{-1} * B$

$A = a * E$

$Enc_{H(k_0)}(m_0)$

$Enc_{H(k_1)}(m_1)$ →

Input: $\sigma \in \{0,1\}$

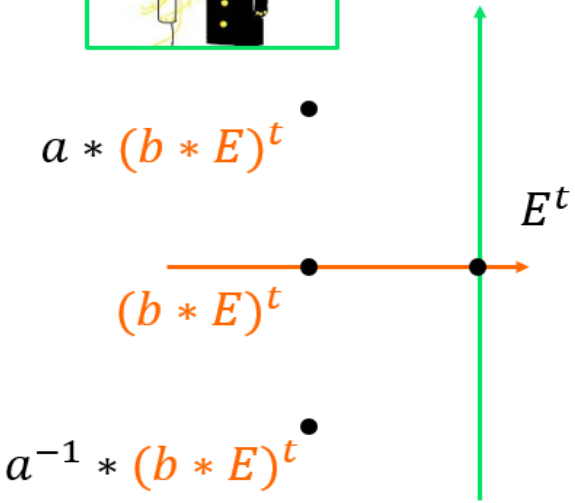
$b \leftarrow Cl$

Set $B = b * E$.

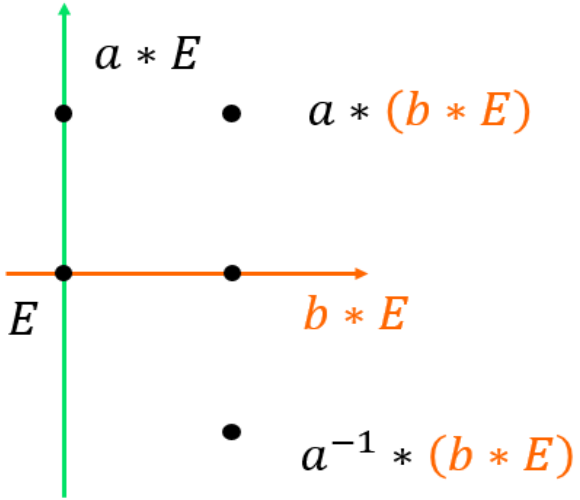
If $\sigma = 1$, set $B = B^t$.

2-Round Construction -Idea

Intuitively, the idea is as like (not a rigorous proof):



E_0



Semi-Simulation

To extract the secret input of the controlled receiver. The main idea here is to use extractablility in ROM and non-commiting encryption to extract the real input. Intuitively, the non-commiting encryption, which can be instantiated by the one-time pad, allows us to produce a dummy ciphertext and produce a corresponding secret key later.

[1] To avoid the delay encryption attack, we add an additional mechanism on improving the one from [BDD+17] to force the adversary to make the random oracle queries of one of the decryption keys.

Semi-Simulation

To extract the secret input of the controlled receiver. The main idea here is to use extractablility in ROM and non-commiting encryption to extract the real input. Intuitively, the non-commiting encryption, which can be instantiated by the one-time pad, allows us to produce a dummy ciphertext and produce a corresponding secret key later.

Hence, assuming the hardness of the reciprocal CSIDH problem, the simulation proceeds as follows,

[1] To avoid the delay encryption attack, we add an additional mechanism on improving the one from [BDD+17] to force the adversary to make the random oracle queries of one of the decryption keys.

Semi-Simulation

To extract the secret input of the controlled receiver. The main idea here is to use extractablility in ROM and non-commiting encryption to extract the real input. Intuitively, the non-commiting encryption, which can be instantiated by the one-time pad, allows us to produce a dummy ciphertext and produce a corresponding secret key later.

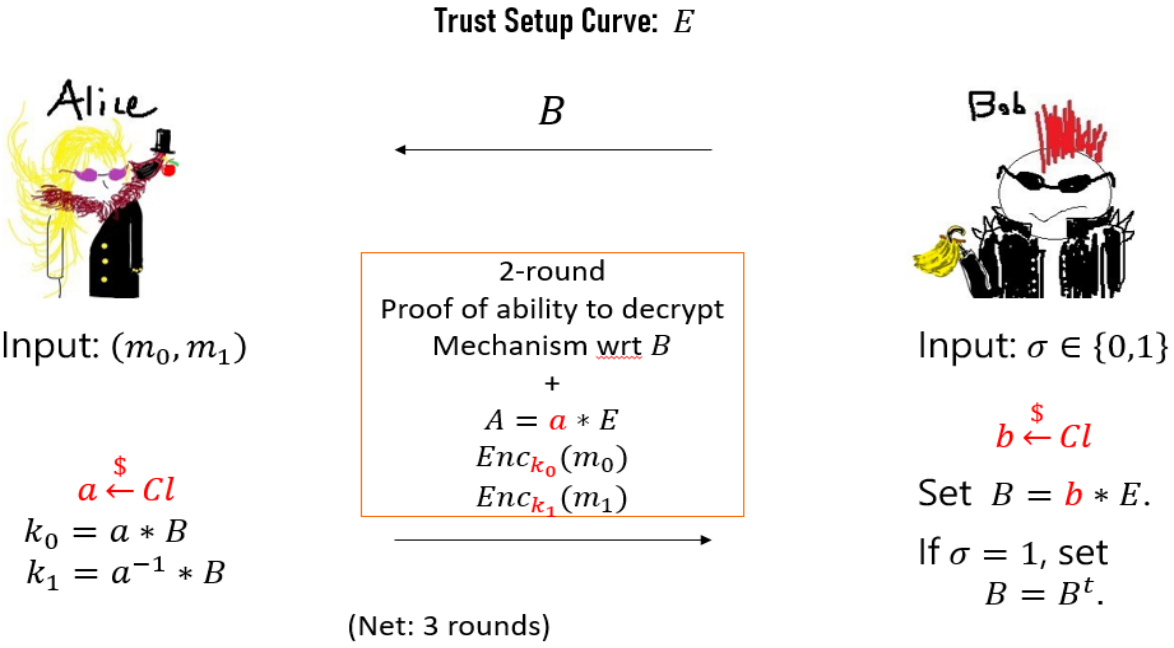
Hence, assuming the hardness of the reciprocal CSIDH problem, the simulation proceeds as follows,

1. Use the non-commiting encryption to produce two dummy ciphertexts when simulating an honest sender.
2. Observe the random oracle queries[1].
3. Once the query of one of k_0, k_1 is made, reply with the scheme-produced key, and the extraction is complete.

[1] To avoid the delay encryption attack, we add an additional mechanism on improving the one from [BDD+17] to force the adversary to make the random oracle queries of one of the decryption keys.

Correction

Recently, a discussion with the third author indicates there is a fixable bug in the paper.



Merging the mechanism and the ciphertext is sufficient if the distinguishing security notion is a non-abort version.

Correction

If abort is part of the output of the functionality, the receiver should show the ability to decrypt before obtaining the ciphertext.

Trust Setup Curve: E

Input: (m_0, m_1)

$a \xleftarrow{\$} Cl$

$k_0 = a * B$

$k_1 = a^{-1} * B$

B

2-round
Proof of ability to decrypt
Mechanism wrt B

$A = a * E$

$Enc_{k_0}(m_0)$

$Enc_{k_1}(m_1)$

(Net: 4 rounds)

Input: $\sigma \in \{0,1\}$

$b \xleftarrow{\$} Cl$

Set $B = b * E$.

If $\sigma = 1$, set
$B = B^t$.

Quadratic Twist Trapdoor

To have UC-security, we need to extract the secret input of the adversary controlling the sender. The key idea here is to setup a trapdoored reciprocal CSIDH problem:

Quadratic Twist Trapdoor

To have UC-security, we need to extract the secret input of the adversary controlling the sender. The key idea here is to setup a trapdoored reciprocal CSIDH problem:

Trapdoor Setup:

1. td $\leftarrow Cl$
2. **Return**: $(\text{td} * E_0, \text{td})$

Quadratic Twist Trapdoor

To have UC-security, we need to extract the secret input of the adversary controlling the sender. The key idea here is to setup a trapdoored reciprocal CSIDH problem:

Trapdoor Setup:

1. $\mathsf{td} \leftarrow Cl$
2. **Return**: $(\mathsf{td} * E_0, \mathsf{td})$

Reciprocal CSIDH Problem Solver
when the public curve $E = \mathsf{td} * E_0$

1. Commit to $X = b * E$ where $b \leftarrow Cl$
2. Receive, say, E' from the challenger
3. **Return**: $(b * E', \mathsf{td}b * (\mathsf{td}^{-1} * (E'))^t)$

Quadratic Twist Trapdoor

To have UC-security, we need to extract the secret input of the adversary controlling the sender. The key idea here is to setup a trapdoored reciprocal CSIDH problem:

Trapdoor Setup:

1. $\mathsf{td} \leftarrow Cl$
2. **Return**: $(\mathsf{td} * E_0, \mathsf{td})$

Reciprocal CSIDH Problem Solver
when the public curve $E = \mathsf{td} * E_0$

1. Commit to $X = b * E$ where $b \leftarrow Cl$
2. Receive, say, E' from the challenger
3. **Return**: $(b * E', \mathsf{td}b * (\mathsf{td}^{-1} * (E'))^t)$

Correctness:
Say $E' = a * E$. Since $X = b * E$, it suffices to show
$\mathsf{td} * (\mathsf{td}^{-1} * (E'))^t = a^{-1} * E$.
By using propositions, we have

$$LHS = \mathsf{td} * (\mathsf{td} * (\mathsf{td}a)^{-1} * E_0) = RHS. \quad \square$$

The Other Half Simulation

Public Curve: E

B

Input: (m_0, m_1)

$A \in \mathcal{E}$

$Enc_{H(k_0)}(\widetilde{m_0})$

$Enc_{H(k_1)}(\widetilde{m_1})$

Input: $\sigma \in \{0,1\}$

1. Setup the trapdoor for the public curve E for the protocol.
2. Invoke the reciprocal CSIDH problem solver (Step 1) to produce B.
3. Upon receiving the curve $A \in \mathcal{E}$ from the controlled sender, invoke the reciprocal CSIDH problem solver to obtain $(k_0', k_1') \in \mathcal{E}^2$.
4. Decrypt the ciphertexts with $H(k_0'), H(k_1')$ and complete the simulation.

Content

Conclusion and Future Work

Proposal	PK_S	PK_R	$\#\ Iso_S$	$\#\ Iso_R$	rounds	Adv Model
[dSGOPS18] I	1	1	3	2	2	Semi-honest
[dSGOPS18] II	3	1	5	2	3	Semi-honest
[Vit19]	2	1	4	2	3	Semi-honest
[AFMP20] I	2λ	2	4λ	$\lambda + 2$	2	Malicious
This work	1	1	3	2	2	Semi-honest
This work	1	1	5	4	4 [2]	Malicious

Table 1: Comparison between isogeny-based OTs on efficiency with respect to the number of elliptic curves in the communication/ the number of isogeny computations for the sender and the receiver. The security parameter is denoted by λ.

Moreover, the underlying assumption, the reciprocal CSIDH problem, is as hard as the computational CSIDH problem.

[2] A correction corresponding to the previous change.

Conclusion and Future Work

Efficient: taking only a constant number of isogeny computations.

1. (Minor) Can we have a quantum-friendly reductions between the reciprocal CSIDH problem and the CSIDH problem?
2. Can we have a round-optimal efficient isogeny-based OT?
3. Can we have an efficient adaptively UC-secure isogeny-based OT?



Thanks for you listening!