A Framework for Universally Composable Oblivious Transfer from One-Round Key-Exchange

Pedro Branco*[1], Jintai Ding[2], Manuel Goulão[1], and Paulo Mateus[1]

[1]SQIG-IT / IST-University of Lisbon
[2]University of Cincinnati

Abstract

Oblivious transfer is one of the main pillars of modern cryptography and plays a major role as a building block for other more complex cryptographic primitives. In this work, we present an efficient and versatile framework for oblivious transfer (OT) using one-round key-exchange (ORKE), a special class of key exchange (KE) where only one message is sent from each party to the other. Our contributions can be summarized as follows:

- We carefully analyze ORKE schemes and introduce new security definitions. Namely, we introduce a new class of ORKE schemes, called Alice-Bob one-round key-exchange (A-B ORKE), and the definitions of message and key indistinguishability.
- We show that OT can be obtained from A-B ORKE schemes fulfilling message and key indistinguishability. We accomplish this by designing a new efficient, versatile and universally composable framework for OT in the Random Oracle Model (ROM). The efficiency of the framework presented depends almost exclusively on the efficiency of the A-B ORKE scheme used since all other operations are linear in the security parameter. Universally composable OT schemes in the ROM based on new hardness assumptions can be obtained from instantiating our framework.

Examples are presented using the classical Diffie-Hellman KE, RLWE-based KE and Supersingular Isogeny Diffie-Hellman KE.

1 Introduction

Oblivious transfer (OT), introduced in the 80s by Rabin [Rab81], is one of the main pillars of modern cryptography. It involves two parties: the sender, which

*Email: pmbranco@math.tecnico.ulisboa.pt

receives as input two messages M_0 and M_1, and the receiver, which receives as input a bit $b \in \{0, 1\}$. Security for the receiver is guaranteed if the sender gets no information on b whereas security for the sender is guaranteed if the receiver gets M_b but no information on M_{1-b}. Despite its simplicity, OT can be employed as a building block to construct other more complex cryptographic primitives [Kil88], such as secure multiparty computation (MPC) [Yao86], privacy-preserving keyword search [FIPR05], or private information retrieval [KO97]. However, for practical purposes, the efficiency and number of OT executions needed to perform these tasks, specially MPC protocols, is a clear bottleneck, even using optimizations, such as OT extensions [ALSZ17]. Hence, the development of efficient OT protocols is crucial to make MPC protocols ubiquitous, which is the motivation of this work.

Recall that key-exchange (KE) allows two parties, usually called Alice and Bob, to share a key while preventing an eavesdropper to get any information about the key. It is probably the oldest public-key cryptographic primitive and its study goes back to the seminal paper of Diffie and Hellman [DH76]. In this work, we consider a special type of KE, called one-round key-exchange (ORKE), where only one message is sent from each party to the other (see, for example, [JKL04, BJS15]). That is, to share a key using an ORKE scheme, Alice sends one message to Bob and Bob sends one message back.

Our main result is the construction of a new efficient and Universally Composable [Can01] framework for OT from ORKE, with very low overhead. Since, it is impossible to achieve universally composable OT in the plain model [CF01], our framework is proven secure in the random oracle model (ROM). In order to design the framework, we analyze carefully ORKE schemes to understand which are the additional conditions required to construct OT.

1.1 Related work

Although it is quite difficult to come up with UC-secure OT protocols, several proposals have been made in recent years. We highlight the ones which use public-key encryption (PKE) schemes as a building block [PVW08, DNMQ12, DDN14, BDD$^+$17, BPRS17] (an idea firstly presented in [BM90]). However, the use of PKE schemes to perform OT is, in most cases, too inefficient, especially in a scenario where one has to create a new pair of public and secret keys for each execution of the OT. This also affects the communication complexity since, for example, post-quantum PKE schemes have very large public keys, and all the above mentioned OT protocols require a public key to be sent from one party to the other.

In real life applications, key exchange (KE) schemes are often used to exchange keys in order to securely communicate using a symmetric-key encryption (SKE) scheme. Hence, the idea of using KE and SKE schemes to design OT protocols seems to follow naturally. This idea was introduced in [CO15] and applied in [HL17, DKLas18], using the Diffie-Hellman protocol and, later, post-quantum versions were presented using the Supersingular Isogeny Diffie-Hellman (SIDH) protocol [BOB18] and using RLWE-based KE [BDGM18]. The main advan-

tage of using KE over PKE schemes to construct OT is that exchanging keys and communicating via SKE is usually much more efficient than communicating using PKE schemes (many times, the decryption algorithm of PKE schemes is much more inefficient than the encryption algorithm. By using KE and SKE over PKE, we avoid the use of the decryption algorithm of PKE). We remark that both the schemes [CO15, BOB18] do not provide UC-security.

1.2 Our contribution

A-B ORKE. Our first contribution is the definition of a new class of key-exchange protocols, called *Alice-Bob one-round key-exchange* (A-B ORKE). An A-B ORKE is an ORKE where the message sent by one party (say Bob) might depend on the message previously sent by the other party (Alice). The specific case when Bob's message does not depend on Alice's is the standard ORKE (as in [BJS15]). Thus, it is obvious that ORKE is a particular case of A-B ORKE. More precisely, it is the non-interactive case of A-B ORKE. To encompass more instances of our OT framework, we work with A-B ORKE schemes. However, we remark that any vanilla ORKE scheme with the same security properties can also be used in our construction.

We introduce new security definitions for A-B ORKE: i) key indistinguishable, meaning that, if Alice sends a uniformly random message (instead of message computed using her secret key), then the shared key computed by Bob is indistinguishable from a uniformly chosen value to her; and ii) message indistinguishable, which means that Alice's message should be indistinguishable from a uniformly random value from Bob's point-of-view. These concepts cannot be trivially defined in a formal way since Alice's message may be composed by several smaller messages, some of them indistinguishable from uniformly random values, but others not.

Hence, consider the set of messages $\mathcal{M}$ that can be computed by Alice. Suppose that there is a group $\mathbb{M}$ and a set $\overline{\mathcal{M}}$ such that $\mathcal{M} \subseteq \overline{\mathcal{M}}$ and consider a group action $\psi : \overline{\mathcal{M}} \times \mathbb{M} \to \overline{\mathcal{M}}$. We define message indistinguishability to be the incapability of an adversary in distinguishing $m \in \mathcal{M}$ from $m' = \psi(m, h)$ where h was sampled uniformly from $\mathbb{M}$. Key indistinguishability is defined similarly as the infeasibility of a adversary to distinguish a key computed by Bob using m' sent by Alice (instead of m) from a uniformly chosen value.

A New framework for OT. As our main result, we present a new framework for building an OT from any A-B ORKE scheme. Our construction shows that it is possible to construct OT from an A-B ORKE that is message and key indistinguishable. The framework is proven to be universally composable in the Random Oracle Model (ROM).

The framework has four rounds and it is extremely efficient, as the overhead is very low. It only requires: i) three messages of the A-B ORKE scheme to be created, and exchanged; ii) a challenge that takes linear-time to create in the security parameter; iii) and two ciphertexts of SKE to be exchanged. Thus,

the efficiency of the framework depends almost exclusively on the A-B ORKE scheme used.

Comparing with other recently proposed frameworks [PVW08, BDD+17], we conclude our construction is more efficient, since we rely on ORKE and SKE, while they rely on PKE schemes. Moreover, our framework can be used to create OT protocols based on hard problems which cannot be achieved when using the frameworks of [PVW08, BDD+17] (such as Supersingular Isogeny-based OT), making our proposal extremely versatile.

Concretely, in our OT framework, the sender and the receiver use, in an ingenious way, the A-B ORKE such that the sender is able to compute two keys k_0 and k_1, one of them (k_b) shared with the receiver. The messages are encrypted by the sender with these keys using a SKE scheme and the receiver can only decrypt one of them.

More precisely, the receiver computes m_{R}^b, where b is its input, and chooses a random seed t. It computes m_{R}^{1-b} such that $m_{\mathsf{R}}^1 = \psi(m_{\mathsf{R}}^0, \mathsf{H}_1(t))$, where H_1 is a random oracle with range $\mathbb{M}$, and sends t and m_{R}^0 to the sender. The sender recovers m_{R}^1 from both t and m_{R}^0 and computes two shared keys. The messages M_0 and M_1 are encrypted using these keys. A challenge has to be sent from the sender to the receiver in order to guarantee security in the UC-framework (we give a detailed explanation for this in the next paragraph). As mentioned before, this paradigm was firstly used to design the simplest OT protocol [CO15]. However, unlike [CO15, BOB18], in our protocol the interaction starts with the receiver which allows to save one round. We remark that this strategy was already used in [BDGM18].

By now, it is well-known that the protocol of [CO15] does not guarantee security in the UC-framework due to subtle timing attacks [BDD+17, BPRS17, DKLas18] (in particular, it does not guarantee security against a corrupted receiver). Here, to achieve UC-security, we have to extract the inputs when only the sender and when only the receiver are corrupted. The extraction of the inputs of the sender is done by programming the random oracle in such a way that the simulator will be able to compute both keys. The extraction of the input of the receiver is a more subtle problem. We solve it using a similar strategy as the one used in [BDD+17], where a challenge is made to the receiver in such a way that it needs to ask the random oracle for information that reveals the input bit to the simulator. To this end, we add two more rounds where a challenge is sent from the sender to the receiver, which can only answer correctly if it queries the random oracle. These queries will be fundamental to design the simulator and avoid attacks from a corrupted receiver. These two extra rounds do not reveal any information about the bit b to the sender since the receiver is able to answer correctly to the challenge no matter its input. On the other hand, the receiver only gets information on the outputs of the secret keys by a random oracle. Observe that the output of the secret key k_{1-b} by the random oracle is not correlated at all with the key. Thus, the receiver does not get any information on the key it does not have.

1.3 Organization of the paper

In Section 2, notation and definitions are presented. The framework for OT is presented in Section 3. The security of the framework is proven in Section 4 and its efficiency is presented in Section 5. Examples are presented in Section 6 using the classical Diffie-Hellman KE, a RLWE-based KE and the Supersingular Isogeny Diffie-Hellman KE.

2 Preliminaries

If $\mathcal{A}$ is an algorithm, we denote by $y \leftarrow \mathcal{A}(x)$ the output of the experiment of running $\mathcal{A}$ on input x. If S is a set and χ is a probabilistic distribution over S, we denote by $x \leftarrow_{\$} S$ the experiment of choosing uniformly at random an element x from S and by $x \leftarrow_{\$} \chi$ the experiment of choosing x from S according to χ. If x and y are two binary strings, we denote by $x|y$ their concatenation and by $x \oplus y$ their bit-wise XOR. If X and Y are two probability distributions, $X \approx Y$ means that they are computationally indistinguishable. A negligible function $\mathsf{negl}(n)$ is a function such that $\mathsf{negl}(n) < 1/\mathsf{poly}(n)$ for every polynomial $\mathsf{poly}(n)$ and sufficiently large n. By a PPT algorithm we mean a probabilistic polynomial-time algorithm. By $\mathsf{Pr}[A : B_1, \ldots, B_n]$ we mean the probability of event A given that events $B_1, \ldots, B_n$ happened sequentially. Throughout this work, the security parameter will be denoted by κ.

In this work, we use symmetric-key encryption schemes. Below, we present the definition of a symmetric-key encryption scheme.

Definition 1. A *symmetric-key encryption* (SKE) scheme $\Delta = (\mathsf{Enc}_\Delta, \mathsf{Dec}_\Delta)$ is a pair of algorithms such that:

- $c \leftarrow \mathsf{Enc}_\Delta(k, M; r)$ is a PPT algorithm that takes as input a shared key k, a message to encrypt M and randomness r and outputs a ciphertext c. Whenever r is omitted, it means that it was chosen uniformly at random;
- $M/\perp \leftarrow \mathsf{Dec}_\Delta(k, c)$ is a PPT algorithm that takes as input a key k and a ciphertext c and outputs a message M, if c was encrypted using k, or an error message $\perp$, otherwise.

A SKE scheme must be sound, that is, $M \leftarrow \mathsf{Dec}_\Delta\left(k, \mathsf{Enc}_\Delta\left(k, M; r\right)\right)$ for any message M and any r. Also, it should be secure, that is, it should be infeasible for an adversary, without knowledge of the secret key, to recover a message from its ciphertext. The security notion that we consider in this work is the one of IND-CPA secure, which is the weakest (and meaningful) security definition that one can consider.

Let Δ be a SKE and consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$: i) $\mathcal{C}$ creates a key k. ii) $\mathcal{A}$ has access to an encryption oracle that it can query a polynomial number of times. iii) At some point, $\mathcal{A}$ outputs two messages M_0 and M_1. iv) A bit b is chosen uniformly by $\mathcal{C}$ and it encrypts M_b and returns the corresponding ciphertext to $\mathcal{A}$. v) Again, $\mathcal{A}$ has access to

an encryption oracle that it can query a polynomial number of times. vi) The game ends with $\mathcal{A}$ outputting a bit b'.

We say that Δ is IND-CPA secure if the advantage of $\mathcal{A}$ in the game above, defined as $\left|\Pr\left[b = b'\right] - \frac{1}{2}\right|$, is negligible in the security parameter.

2.1 UC-security and ideal functionalities

The Universal Composability (UC) framework, firstly introduced by Canetti [Can01], allows us to analyze the security of protocols, not just *per se*, but also when composed with other protocols. Due to the lack of space, only a brief introduction on the UC-framework is presented. For more details on this subject we refer the reader to [Can01].

In a nutshell, to prove UC security of a protocol π (usually called the real-world execution) one compares it to an ideal version of the primitive, defined *a priori* (usually called the ideal-world execution). The entities involved in the ideal-world execution are dummy parties which interact via an ideal functionality $\mathcal{F}$. These dummy parties may or may not be corrupted by an ideal adversary Sim, usually called the simulator. The functionality works as a trusted party: it receives inputs from all the entities involved and returns to each one something, depending on the primitive being implemented. In this way, each of the parties learns nothing but its own input and output. In the real-world execution, several parties interact between them via some protocol π, which implements the desired primitive. These parties may or may not be corrupted by some adversary $\mathcal{A}$. An entity $\mathcal{E}$, often called the environment, oversees the executions in both the ideal and the real worlds. At the end of the executions, the environment is asked to distinguish them. The intuition of the UC-framework is that a protocol π is secure if the environment $\mathcal{E}$ is not able to distinguish the real-world execution of π from the ideal-world execution of $\mathcal{F}$. If this happens, we can conclude that a real-world adversary $\mathcal{A}$ does not have more power than an ideal-world adversary Sim. Hence, whatever strategy a real-world adversary $\mathcal{A}$ uses to cheat in the execution of π, it can also be used by an ideal-world adversary Sim. Since we define the ideal functionality in order to avoid attacks from any adversary, we can conclude that there is no strategy for the real-world adversary $\mathcal{A}$ that allows it to know more than its own input and output.

Formally, let π be a protocol where n parties and an adversary $\mathcal{A}$ are involved. We denote the output of the environment $\mathcal{E}$ in the end of the real-world execution of π with adversary $\mathcal{A}$ by $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{E}}$. The output of $\mathcal{E}$ at the end of the ideal-world execution of a functionality $\mathcal{F}$ with adversary Sim is denoted by $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$. The following definition introduces the notion of a protocol emulating (in a secure way) some ideal functionality.

Definition 2. We say that a protocol π *UC-realizes* $\mathcal{F}$ if for every PPT adversary $\mathcal{A}$ there is a PPT simulator Sim such that for all PPT environments $\mathcal{E}$,

$$\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}} \approx \mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{E}}$$

where $\mathcal{F}$ is an ideal functionality.

Oblivious transfer (OT), firstly introduced by Rabin [Rab81], is a crucial primitive in cryptography. We describe the $\binom{2}{1}$-OT ideal functionality $\mathcal{F}_{\mathrm{OT}}$, as presented in [CLOS02]. Let $\lambda \in \mathbb{N}$ be a fixed value known to both parties, $M_0, M_1 \in \{0,1\}^\lambda$ and $b \in \{0,1\}$. The value sid represents the session ID and the ID of the parties involved in the protocol.

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

Parameters: $\mathsf{sid}, \lambda \in \mathbb{N}$ known to both parties.

- Upon receiving (sid, M_0, M_1) from S, $\mathcal{F}_{OT}$ stores M_0, M_1 and ignores future messages from S with the same sid;
- Upon receiving (sid, b) from R, $\mathcal{F}_{OT}$ checks if it has recorded (sid, M_0, M_1). If so, returns (sid, M_b) to R and $(\mathsf{sid}, \textit{receipt})$ to S and halts. Else, it sends nothing but continues running.

Unfortunately, it is impossible to design universally composable OT protocols in the plain model, that is, without any setup assumption [CF01]. Hence, we use the random oracle model (ROM) to construct our UC-secure OT protocol. To this end, we work on the $\mathcal{F}_{\mathrm{RO}}$-hybrid model in order to model random oracles in the UC framework. The random oracle ideal functionality $\mathcal{F}_{\mathrm{RO}}$ is presented below.

$\mathcal{F}_{\mathbf{RO}}$ functionality

Parameters: Let $\mathcal{D}$ be the range of $\mathcal{F}_{\mathrm{RO}}$ and L be a list initially empty.

- Upon receiving a query (sid, q) from a party $\mathcal{P}$ or from an adversary $\mathcal{A}$, $\mathcal{F}_{\mathrm{RO}}$ proceeds as follows:
  - If there is a pair $(q, h) \in L$, it returns (sid, h);
  - Else, it chooses $h \leftarrow_{\$} \mathcal{D}$, stores the pair $(q, h) \in L$ and returns (sid, h).

The idea behind the $\mathcal{F}_{\mathrm{RO}}$-hybrid model is that every party involved in both the ideal-world execution of $\mathcal{F}$ and the real-world execution of the protocol π (including the adversary) have access to an ideal functionality $\mathcal{F}_{\mathrm{RO}}$, which behaves as a random oracle. The environment can access this ideal functionality through the adversary. We denote by $\mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi,\mathcal{A},\mathcal{E}}$ the output of the environment after the real-world execution of the protocol π with an adversary $\mathcal{A}$ in the real-world, with the ideal functionality $\mathcal{F}_{\mathrm{RO}}$. The notion of a protocol securely emulating an ideal functionality can be adapted to this model.

Definition 3. We say that a protocol π *UC-realizes* $\mathcal{F}$ in the *$\mathcal{F}_{RO}$-hybrid model* if for every PPT adversary $\mathcal{A}$ there is a PPT simulator Sim such that for all

PPT environments $\mathcal{E}$,

$$\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi,\mathcal{A},\mathcal{E}}.$$

In this work, we consider static malicious adversaries. That is, an adversary corrupting any of the parties can deviate arbitrarily as it wishes from the protocol. However the parties are corrupted by the adversary before the beginning of the protocol and they remain so until the end of the protocol.

2.2 One-round key-exchange

One-round key-exchange (ORKE) is a cryptographic primitive that allows two parties to agree on a shared key while an eavesdropper gets no information on the key. In an ORKE scheme, only one message is sent from each party to the other. The notion of ORKE scheme appears for the first time in the seminal work of Diffie and Hellman [DH76]. In an ORKE scheme, the messages of Alice and Bob can be computed offline. We present the definition of ORKE. Our definition is a variant of the one presented in [BJS15], since we are not interested in key reuse (which is the motivation of the work of [BJS15]).

Definition 4. A *one-round key-exchange* (ORKE) scheme Π is defined by a tuple of algorithms $(\mathsf{Gen}_\Pi, \mathsf{Msg}_\Pi, \mathsf{Key}_\Pi)$ where:

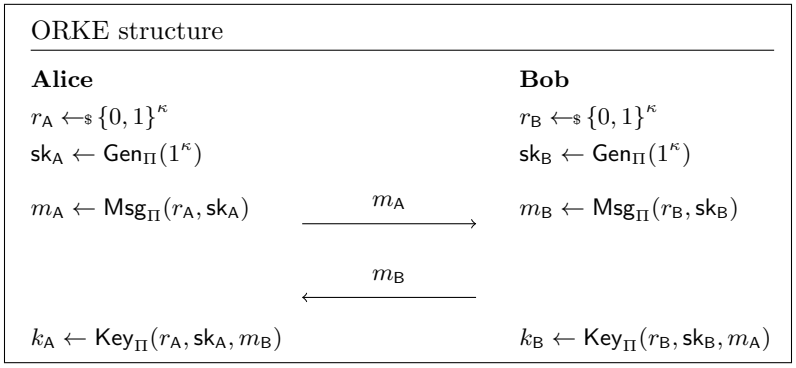
- $\mathsf{sk} \leftarrow \mathsf{Gen}_\Pi(1^\kappa, r)$ is an algorithm that takes as input a security parameter κ and a random value r and outputs a secret keys sk. Whenever r is omitted, it means that it is chosen uniformly at random.
- $m_i \leftarrow \mathsf{Msg}_\Pi(r_i, \mathsf{sk}_i)$ is an algorithm that takes as input a random value r_i and secret key sk_i, and outputs a message m_i.
- $k \leftarrow \mathsf{Key}_\Pi(r_i, \mathsf{sk}_i, m_j)$ is an algorithm that takes as input a secret key sk_i, a random r_i and a message m_j and outputs a key k.

A ORKE scheme Π should be *sound*. That is, if for all $\mathsf{sk}_i \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$ and $\mathsf{sk}_j \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$ and for all $r_i, r_j \leftarrow \{0,1\}^\kappa$, it holds that

$$\mathsf{Key}_\Pi(r_i, \mathsf{sk}_i, m_j) = \mathsf{Key}_\Pi(r_j, \mathsf{sk}_j, m_i)$$

where $m_i \leftarrow \mathsf{Msg}_\Pi(r_i, \mathsf{sk}_i)$ and $m_j \leftarrow \mathsf{Msg}_\Pi(r_j, \mathsf{sk}_j)$.

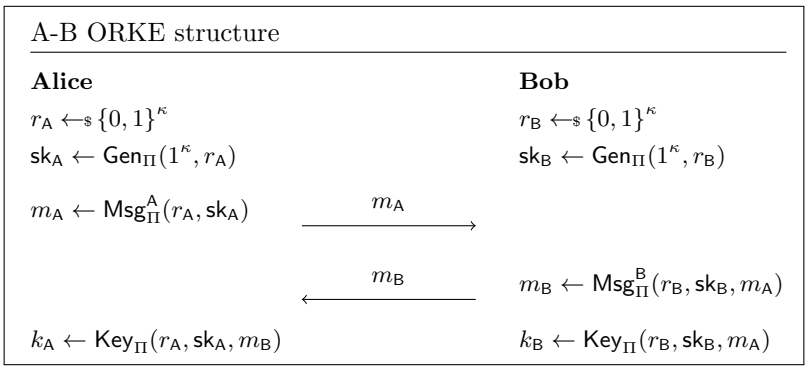
For two parties Alice and Bob to exchange a shared key using Π, they both generate a secret key and run the algorithm Msg_Π with their respective secret key. They send the resulting message to the other party and, finally, they both run the key algorithm Key_Π in order to obtain the shared key.



We want our framework to be as general as possible, so we define a new type of ORKE scheme which we call Alice-Bob one-round key-exchange (A-B ORKE). An A-B ORKE scheme is an ORKE scheme where Alice sends her message m_{A} first. So, Bob's message m_{B} can depend on m_{A}. An A-B ORKE can be seen as a generalization of the concept of ORKE. It is obvious that every ORKE scheme is an A-B ORKE scheme. However, the converse is not true in general: in Ding's KE [DXL12] or in New Hope KE [ADPS16], Bob's message depends on Alice's, and thus, they are A-B ORKE schemes but not ORKE schemes.

Definition 5. An *Alice-Bob one-round key-exchange* (A-B ORKE) scheme Π is defined by three algorithms $(\mathsf{Gen}_{\Pi}, \mathsf{Msg}_{\Pi}, \mathsf{Key}_{\Pi})$, where $\mathsf{Msg}_{\Pi} = (\mathsf{Msg}_{\Pi}^{\mathsf{A}}, \mathsf{Msg}_{\Pi}^{\mathsf{B}})$, such that

- $\mathsf{sk} \leftarrow \mathsf{Gen}_{\Pi}(1^{\kappa}, r)$ is an algorithm that takes as input a security parameter κ and a random value r and outputs a secret key sk.
- $m_i \leftarrow \mathsf{Msg}_{\Pi}^{\mathsf{A}}(r_i, \mathsf{sk}_i)$ is an algorithm that takes as input a random value r_i and secret key sk_i, and outputs a message m_i.
- $m_j \leftarrow \mathsf{Msg}_{\Pi}^{\mathsf{B}}(r_j, \mathsf{sk}_j, m_i)$ is an algorithm that takes as input a random value r_i, and secret key sk_i and a message m_i previously sent by the other party and outputs a message m_j.
- $k \leftarrow \mathsf{Key}_{\Pi}(r_i, \mathsf{sk}_i, m_j)$ is an algorithm that takes as input a secret key sk_i, a random r_i and a message m_j and outputs a key k.



Again, an A-B ORKE scheme should be sound, that is, the key obtained by Alice should be equal to the key obtained by Bob, except with negligible probability.

To design our framework for OT, we need that the A-B ORKE scheme used fulfills certain properties which we have called *message indistinguishability* and *key indistinguishability*.

First, we need to introduce the notion of *non-redundant message for key generation*. Intuitively, a non-redundant message outputted by the algorithm Msg_Π is a message such that every part of it is used to construct the shared key. We define such property for messages sent by Alice:

Definition 6. Let κ be the security parameter, $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Msg}_\Pi, \mathsf{Key}_\Pi)$ be an A-B ORKE scheme and $a \in \mathbb{N}$. Let $m = (m_1, \ldots, m_a) \leftarrow \mathsf{Msg}^A_\Pi(r_A, \mathsf{sk}_A)$, and $m_B \leftarrow \mathsf{Msg}^B_\Pi(r_B, \mathsf{sk}_B, m)$. Let $m' = (m_{i_1}, \ldots, m_{i_b})$ for any proper subset $\{i_1, \ldots, i_b\} = S \subset \{1, \ldots, a\}$. We say that $m = (m_1, \ldots, m_a)$ is *non-redundant for key generation* (NRKG) if

$$\mathsf{Pr}\left[k_B \neq k'_B \wedge k_B = k_A : k_B \leftarrow \mathsf{Key}_\Pi(r_B, \mathsf{sk}_B, m), \right.$$
$$\left. k'_B \leftarrow \mathsf{Key}_\Pi(r_B, \mathsf{sk}_B, m'), k_A \leftarrow \mathsf{Key}_\Pi(r_A, \mathsf{sk}_A, m_B)\right] \geq 1 - \mathsf{negl}(\kappa)$$

for $\mathsf{sk}_A \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $\mathsf{sk}_B \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $r_A, r_B \leftarrow_\$ \{0,1\}^\kappa$.

We introduce the concept of *message indistinguishable* for an A-B ORKE scheme. Again, we define this property for messages sent by Alice.

Recall that, if $(\mathbb{G}, *)$ is a group with operation $*$ and X is a set, then a right group action $\psi : X \times \mathbb{G} \to X$ is a function that satisfies: i) $\psi(x, e) = x$ for every $x \in X$ and the identity element e of $\mathbb{G}$; and ii) $\psi(\psi(x, g), h) = \psi(x, g * h)$ for every $x \in X$ and $g, h \in \mathbb{G}$.

In the following, let $(\mathbb{M}, *)$ be a group and $\mathcal{M}$ be the space of non-redundant messages for key generation outputted by the algorithm Msg^A_Π. Let $\psi : \overline{\mathcal{M}} \times (\mathbb{M}, *) \to \overline{\mathcal{M}}$ be a right group action of $(\mathbb{M}, *)$ on $\overline{\mathcal{M}}$, where $\overline{\mathcal{M}}$ is a set such that $\mathcal{M} \subseteq \overline{\mathcal{M}}$.

Definition 7. Let κ be the security parameter, $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Msg}_\Pi, \mathsf{Key}_\Pi)$ be an A-B ORKE scheme that is NRKG and ψ, $\mathbb{M}$ and $\mathcal{M}$ be as described above. We

say that an A-B ORKE protocol is *ψ-message indistinguishable* if, for any PPT adversary $\mathcal{A}$, we have that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A}(x, \mathsf{sk_B}, h) : x \leftarrow \mathsf{Msg}_\Pi^{\mathsf{A}}(r_\mathsf{A}, \mathsf{sk_A}), h \leftarrow_\$ \mathbb{M}\right]$$
$$- \mathsf{Pr}\left[1 \leftarrow \mathcal{A}(x, \mathsf{sk_B}, h) : y \leftarrow \mathsf{Msg}_\Pi^{\mathsf{A}}(r_\mathsf{A}, \mathsf{sk_A}), h \leftarrow_\$ \mathbb{M}, x \leftarrow \psi(y, h)\right] \leq \mathsf{negl}(\kappa)$$

where $\mathsf{sk_A} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $\mathsf{sk_B} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $r_\mathsf{A} \leftarrow_\$ \{0,1\}^\kappa$.

The intuition behind this definition is that we need $x = \psi(y, h)$ to be indistinguishable from uniformly chosen elements from some set, where $y \in \mathcal{M}$ and h is an element in a set $\mathbb{M}$. We also need h to have inverse in $\mathbb{M}$, to be able to recover y. One possible solution is to consider A-B ORKE schemes for which the set $\mathcal{M}$ (the set of outputs of $\mathsf{Msg}_\Pi^{\mathsf{A}}$) forms a group, and consider $\mathcal{M}$ to be $\mathbb{M}$. This happens when we consider the Diffie-Hellman KE, for example. However, there are cases where the set $\mathcal{M}$ may not have inverses or, even worse, it may not be closed under any operation (e.g., consider $\mathcal{M}$ to be the set of LWE samples [Reg05], as in the A-B ORKE schemes of [DXL12, Pei14, ADPS16]). But observe that cases like these LWE-based schemes also have some type of indistinguishability (see Example8). From this example, we conclude that we only need the elements in $\mathcal{M}$ to be indistinguishable from elements in $\overline{\mathcal{M}}$, where $\overline{\mathcal{M}}$ is a set such that $\mathcal{M} \subseteq \overline{\mathcal{M}}$. Again, for the framework to be as general as possible, we define message indistinguishability as the incapability to distinguish elements of $\mathcal{M}$ from elements of $\overline{\mathcal{M}}$. This definition also includes the cases where $m = (m_1, \ldots, m_a) \in \mathcal{M}$ is composed by several smaller messages m_i and where only part of these coordinates are affected by the action of the group $(\mathbb{M}, *)$ (while the other coordinates remain the same).

Example 8. *Consider $\mathcal{M}$ to be the the set of LWE samples in $\mathbb{Z}_q^n = (\mathbb{Z}/q\mathbb{Z})^n$ for some $n \in \mathbb{N}$ and some $q \in \mathbb{N}$, as in several lattice-based A-B ORKE schemes [DXL12, Pei14, ADPS16]. Now consider $\overline{\mathcal{M}}$ and $\mathbb{M}$ to be $\mathbb{Z}_q^n$ and the operation $*$ to be the sum $+$ in $\mathbb{Z}_q^n$. Observe that, when the action $\psi : \overline{\mathcal{M}} \times (\mathbb{Z}_q^n, +) \to \overline{\mathcal{M}}$ is defined as $\psi(x, h) = x + h$, then $\psi(x, h)$ is uniformly chosen at random in $\mathbb{Z}_q^n$ when $h \leftarrow_\$ \mathbb{M}$. From the LWE assumption [Reg05], which states that LWE samples are indistinguishable from uniformly chosen values from $\mathbb{Z}_p^n$, we conclude that the schemes [DXL12, Pei14, ADPS16] are ψ-message indistinguishable. The definition above generalizes this notion of indistinguishability of messages.*

Finally, we also need that the key obtained by Bob is indistinguishable from a uniformly chosen value, when it is given a uniformly chosen value instead of the message obtained by Alice when running $\mathsf{Msg}_\Pi^{\mathsf{A}}$.

Definition 9. Let κ be the security parameter, $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Msg}_\Pi, \mathsf{Key}_\Pi)$ be an A-B ORKE scheme that is NRKG, ψ, $\mathbb{M}$ and $\mathcal{M}$ be as above and $\mathcal{K} = \{0,1\}^\beta$, where β is the length of the key outputted by the Key_Π algorithm. We say that

an A-B ORKE protocol is *ψ-key indistinguishable* if, for any PPT adversary $\mathcal{A}$, we have that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A}(k, \mathsf{sk_A}, m, m', h) : k \leftarrow \mathsf{Key}_\Pi(r_\mathsf{B}, \mathsf{sk_B}, m')\right] - \mathsf{Pr}[1 \leftarrow \mathcal{A}(k, \mathsf{sk_A}, m, m', h) : k \leftarrow_\$ \mathcal{K}] \leq \mathsf{negl}(\kappa)$$

where $\mathsf{sk_A} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $\mathsf{sk_B} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, $m' \leftarrow \psi(m, h)$ with $m \leftarrow \mathsf{Msg}_\Pi^\mathsf{A}(r_\mathsf{A}, \mathsf{sk_A})$ and $h \leftarrow_\$ \mathbb{M}$.

As an example, KE protocols fulfilling AM security, as in [CK01], also fulfill key indistinguishability. However, we remark that AM security is stronger than key indistinguishability, since the former notion also provides some form of composability.

Examples of A-B ORKE schemes that fulfill both of these conditions are Diffie-Hellman [DH76], the lattice-based protocols of [DXL12, ADPS16], and the Supersingular Isogeny Diffie-Hellman [JDF11] (we discuss these cases in Section 6).

3 A framework for OT using ORKE

In this section, we present the framework for OT. Let κ be the security parameter. Let $\overline{\mathcal{M}}$ and $\psi : \overline{\mathcal{M}} \times (\mathbb{M}, *) \to \overline{\mathcal{M}}$ be the right group action as defined in Section 2.2, where $\mathcal{M}$ is the set of outputs of algorithm $\mathsf{Msg}_\Pi^\mathsf{A}$ and let $(\mathbb{M}, *)$ be a group. We assume that, given $x, y \in \overline{\mathcal{M}}$, it is computationally easy to find $h \in \mathbb{M}$ such that $x \leftarrow \psi(y, h)$. Let $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Msg}_\Pi, \mathsf{Key}_\Pi)$ be an A-B ORKE protocol that is ψ-message indistinguishable and ψ-key indistinguishable, and $\Delta = (\mathsf{Enc}_\Delta, \mathsf{Dec}_\Delta)$ be an IND-CPA secure symmetric-key encryption protocol. Suppose that the sender S wants to obliviously send M_0 and M_1, and that the receiver R wants to receive the message M_b, where $b \in \{0, 1\}$ is its input. Both S and R start by generating a secret key, $\mathsf{sk_S} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$ and $\mathsf{sk_R} \leftarrow \mathsf{Gen}_\Pi(1^\kappa)$, respectively.

Let H_i, for $i = 1, \ldots, 4$ be four different instances of the random oracle functionality $\mathcal{F}_{\mathrm{RO}}$. More precisely, $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{M}$ is used to create a random message from a honestly created message (for the receiver), $\mathsf{H}_2 : \{0,1\}^* \to \mathcal{K} = \{0,1\}^\beta$ where β is the size of the keys outputted by the Key_Π algorithm, and $\mathsf{H}_3 : \{0,1\}^* \to \{0,1\}^{2\kappa+\beta}$ and $\mathsf{H}_4 : \{0,1\}^* \to \{0,1\}^\kappa$ for a challenge-response interaction.

The framework. The scheme has four communication rounds and the receiver R sends the first message.

1. When activated with its input, the receiver R:
   - Chooses at random $t, r_\mathsf{R} \leftarrow_\$ \{0,1\}^\kappa$;

- Queries H_1 on (sid, t) and sets the output to $h \in \mathbb{M}$;
- Computes $m^b_{\mathsf{R}} \leftarrow \mathsf{Msg}^{\mathsf{A}}_{\Pi}(r_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$;
- If $b = 1$, it computes $m^0_{\mathsf{R}} \leftarrow \psi(m^1_{\mathsf{R}}, h^{-1})$. Else, it continues;
- Sends $(\mathsf{sid}, t, m^0_{\mathsf{R}})$ to S.

2. Upon receiving $(\mathsf{sid}, t, m^0_{\mathsf{R}})$ from R, the sender S:

- Chooses $r_{\mathsf{S}} \leftarrow \{0,1\}^{\kappa}$;
- Queries H_1 on (sid, t) and sets the output to $h' \in \mathbb{M}$;
- Computes $m^1_{\mathsf{R}} \leftarrow \psi(m^0_{\mathsf{R}}, h')$;
- Computes $m^0_{\mathsf{S}} \leftarrow \mathsf{Msg}^{\mathsf{B}}_{\Pi}(r_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, m^0_{\mathsf{R}})$ and $m^1_{\mathsf{S}} \leftarrow \mathsf{Msg}^{\mathsf{B}}_{\Pi}(r_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, m^1_{\mathsf{R}})$;
- Computes the keys $k^0_{\mathsf{S}} \leftarrow \mathsf{Key}_{\Pi}(r_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, m^0_{\mathsf{R}})$ and $k^1_{\mathsf{S}} \leftarrow \mathsf{Key}_{\Pi}(r_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, m^1_{\mathsf{R}})$;
- Chooses $w_0, z_0, w_1, z_1 \leftarrow \{0,1\}^{\kappa}$;
- Queries H_2 on $(\mathsf{sid}, k^0_{\mathsf{S}})$ setting the output to $\bar{k}^0_{\mathsf{S}}$, and on $(\mathsf{sid}, k^1_{\mathsf{S}})$ setting the output to $\bar{k}^1_{\mathsf{S}}$;
- Queries H_3 on (sid, w_0) setting the output to $\bar{w}_0$, and on (sid, w_1) setting the output to $\bar{w}_1$;
- Computes $a_0 \leftarrow \mathsf{Enc}_{\Delta}(\bar{k}^0_{\mathsf{S}}, w_0; z_0)$ and $a_1 \leftarrow \mathsf{Enc}_{\Delta}(\bar{k}^1_{\mathsf{S}}, w_1; z_1)$;
- Sets $u_0 \leftarrow \bar{w}_0 \oplus (w_1 | \bar{k}^1_{\mathsf{S}} | z_1)$ and $u_1 \leftarrow \bar{w}_1 \oplus (w_0 | \bar{k}^0_{\mathsf{S}} | z_0)$;
- Queries H_4 on $(\mathsf{sid}, w_0, w_1, z_0, z_1)$ setting the output to ch;
- Sends $(\mathsf{sid}, m^0_{\mathsf{S}}, m^1_{\mathsf{S}}, a_0, a_1, u_0, u_1)$ to R.

3. Upon receiving $(\mathsf{sid}, m^0_{\mathsf{S}}, m^1_{\mathsf{S}}, a_0, a_1, u_0, u_1)$ from S, the receiver R:

- Computes $k_{\mathsf{R}} \leftarrow \mathsf{Key}_{\Pi}(r_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}, m^b_{\mathsf{S}})$;
- Queries H_2 on $(\mathsf{sid}, k_{\mathsf{R}})$ setting the output to $\bar{k}_{\mathsf{R}}$;
- Decrypts $x_b \leftarrow \mathsf{Dec}_{\Delta}(\bar{k}_{\mathsf{R}}, a_b)$;
- Queries H_3 on (sid, x_b) setting the output to $\bar{x}_b$;
- Computes $(x_{1-b} | \bar{k}^{1-b}_{\mathsf{R}} | y_{1-b}) = u_b \oplus \bar{x}_b$;
- Queries H_3 on (sid, x_{1-b}) setting the output to $\bar{x}_{1-b}$;
- Recovers $(x'_b | \bar{k}^b_{\mathsf{R}} | y_b) = u_{1-b} \oplus \bar{x}_{1-b}$;
- Checks if $a_0 = \mathsf{Enc}_{\Delta}(\bar{k}^0_{\mathsf{R}}, x_0; y_0)$, if $a_1 = \mathsf{Enc}_{\Delta}(\bar{k}^1_{\mathsf{R}}, x_1; y_1)$, if $\bar{k}^b_{\mathsf{R}} = \bar{k}_{\mathsf{R}}$ and if $x'_b = x_b$. It aborts if any of these conditions fail;
- Queries H_4 on $(\mathsf{sid}, x_0, x_1, y_0, y_1)$ and sets the output to ch';
- Sends (sid, ch') to S.

4. Upon receiving (sid, ch') from R, the sender S:
   - Checks if $ch = ch'$. It aborts, if the test fails;
   - Encrypts $c_0 \leftarrow \mathsf{Enc}_\Delta(k^0_\mathsf{S}, M_0)$ and $c_1 \leftarrow \mathsf{Enc}_\Delta(k^1_\mathsf{S}, M_1)$;
   - Sends (sid, c_0, c_1) to R and halts.
5. Upon receiving (sid, c_0, c_1) from S, the receiver R:
   - Decrypts $M_b \leftarrow \mathsf{Dec}_\Delta(k_\mathsf{R}, c_b)$;
   - Outputs M_b and halts.

We call this framework π_{OT}. In the first two rounds, a key exchange is used in a ingenious way that allows the sender and the receiver to share a common key such that: i) the sender does not know which of the two keys it has computed is shared with the receiver; and ii) the receiver has no information about the other key.

In the proof of security in the UC-framework, the extraction of the inputs of the sender (the messages M_0 and M_1) is done by programming the random oracle H_1 in such a way that the simulator has both keys and is able to decrypt both ciphertexts c_0 and c_1.

The challenge that the sender sends to the receiver is necessary to extract the input bit b of the receiver. The extraction is possible when the receiver asks k_R to the random oracle. Here, the simulator is able to know the bit b by comparing this value with the keys the dummy sender has computed. Note that this challenge does not carry any information about the key k^{1-b}_S: the only values that the receiver gets from this challenge are random values x_0, x_1, y_0, y_1 and the output of the secret keys by the random oracle, which, by definition, are completely uncorrelated with the keys.

Extension to $\binom{N}{1}$-OT. It is straightforward to extend the framework above to an $\binom{N}{1}$-OT, where S's input is composed by N messsages $M_0, \ldots M_{N-1}$ and R's input is $b \in \{0, \ldots, N-1\}$ such that R receives M_b.

In the first message, the receiver R, instead of just sending t, sends $t_1, \ldots, t_{N-2}$ along with m^0_R. S computes $h'_i \leftarrow \mathsf{H}_1(t_i)$ and the messages $m^i_\mathsf{R} \leftarrow \psi(m^0_\mathsf{R}, h'_i)$ for $i = 1, \ldots, N-1$. From these messages, S computes N keys such that one of them is shared with R.

Also, S chooses $w_0, \ldots, w_{N-1}, z_0, \ldots, z_{N-1} \leftarrow_\$ \{0,1\}^\kappa$ and sets the challenge to be

$$ch \leftarrow \mathsf{H}_4(\mathsf{sid}, w_0, \ldots, w_{N-1}, z_0, \ldots, z_{N-1}),$$

instead of just $(\mathsf{sid}, w_0, w_1, z_0, z_1)$. Furthermore, S needs to compute

$$a_i \leftarrow \mathsf{SEnc}_\Delta(\bar{\mathsf{sk}}^i_\mathsf{S}, w_i; z_i)$$

and

$$u_i \leftarrow \bar{w}_i \oplus (w_{i+1 \mod N} | \bar{\mathsf{sk}}_{\mathsf{S}}^{i+1 \mod N} | z_{i+1 \mod N})$$

for $i = 0, \ldots, N-1$. Finally, it sends $(a_0, \cdots, a_{N-1}, u_0, \cdots, u_{N-1})$ to R. The remaining steps can be easily adapted from the version presented above.

Security for the receiver. Security for R is guaranteed by the ψ-message indistinguishability of the A-B ORKE scheme used. Note that S receives two messages from R. In the first one, it receives m_{R}^0 (from which it can recover m_{R}^1) but, by the ψ-message indistinguishability property, S has no information on which message was the one computed using the $\mathsf{Msg}_{\Pi}^{\mathsf{A}}$ algorithm and which one is a random value. Thus, it does not know which message R uses to compute its key.

The second message sent by R to S is ch', but note that R can compute ch' regardless of its input, given that S has behaved honestly. Observe that when S does not behave honestly, then R aborts the execution. We conclude that it is infeasible for S to know the input of the receiver.

Security for the sender. The first message that S sends to R is $(m_{\mathsf{S}}^0, m_{\mathsf{S}}^1, a_0, a_1, u_0, u_1)$. By the ψ-key indistinguishability of the A-B ORKE scheme used, R is not able to derive a key from m_{S}^{1-b}. Otherwise, it could break the ψ-key indistinguishability property of the underlying A-B ORKE. Moreover, the only information R gets from a_0, a_1, u_0, u_1 about k_{S}^{1-b} is its output by H_2, that is $\bar{k}_{\mathsf{S}}^{1-b}$. Since H_2 is modeled as a random oracle, the values k_{S}^{1-b} and $\bar{k}_{\mathsf{S}}^{1-b}$ are not correlated.

The second message sent from S to R is composed by the ciphertexts c_0, c_1. Given that the SKE scheme Δ is secure, it is infeasible for R to get information about M_{1-b} if it does not have the corresponding secret key. We conclude that it is infeasible for the receiver to get both messages.

4 Security proof

We prove the main result of this paper which guarantees the UC-security of the proposed OT protocol π_{OT}.

Theorem 10. *The protocol π_{OT} UC-realizes $\mathcal{F}_{OT}$ in the $\mathcal{F}_{RO}$-hybrid model against static malicious adversaries, given that Δ is* IND-CPA *secure and the A-B ORKE scheme used is ψ-message indistinguishable and ψ-key indistinguishable.*

We have to prove that, for every adversary $\mathcal{A}$ corrupting any number of parties in the protocol, there is a simulator such that no environment can distinguish the real-world from the ideal-world executions.

We begin with the trivial case: when the adversary is corrupting both the sender and the receiver then the simulator just runs internally the adversary which generates the messages for both the sender and the receiver.

When the adversary is is not corrupting any party, then the simulator just follows the protocol with the random inputs, forwarding every message to $\mathcal{A}$. Observe that the obtained transcript is indistinguishable from any other transcript (with other inputs) from the point-of-view of $\mathcal{A}$ and, thus, $\mathcal{E}$. Thus, the real-world and the ideal-world executions are indistinguishable in this case.

The proof follows from Lemma 11 and Lemma 12, where the remaining cases are considered.

Lemma 11. *Given any PPT adversary $\mathcal{A}(\mathsf{R})$ corrupting the receiver R, there is a PPT simulator Sim such that for every PPT environment $\mathcal{E}$ we have*

$$\mathsf{IDEAL}_{\mathcal{F}_{OT},\mathsf{Sim},\mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_{RO}}_{\pi_{OT},\mathcal{A}(\mathsf{R}),\mathcal{E}},$$

given that Δ is IND-CPA *secure and the A-B ORKE scheme used is ψ-message indistinguishable and ψ-key indistinguishable.*

Proof. To prove security against a corrupted receiver, we have to construct a simulator that is able to extract the input of a corrupted receiver, given any adversary $\mathcal{A}(\mathsf{R})$ corrupting the receiver. In this case, the input of $\mathcal{A}(\mathsf{R})$ is a bit $b \in \{0,1\}$. Here, the idea of the extraction is that $\mathcal{A}(\mathsf{R})$ has to query the random oracle on its key, thus revealing b to the simulator. Upon extracting the bit b, the simulator can send b to the ideal functionality that will return back a message M_b. To finish the simulation, the simulator follows the protocol π_{OT} and obliviously send M_b and $M_{1-b} \leftarrow 0^\lambda$.

1. Upon activating the adversary, the simulator Sim simulates the random oracles H_1, H_2, H_3 and H_4 in the following way: Sim keeps a list L_i for each H_i, for $i = 1, \ldots, 4$, which is initially empty. Whenever $\mathcal{A}(\mathsf{R})$ queries H_i on (sid, q), Sim checks if there is $(q, h) \in L_i$. If so, it returns h. Else, it chooses h uniformly at random, records the pair (q, h) in L_i and returns h.
2. Upon receiving $(\mathsf{sid}, t, m^0_{\mathsf{R}})$ from the adversary $\mathcal{A}(\mathsf{R})$, the simulator Sim:
   - Follows the protocol and sends $(\mathsf{sid}, m^0_{\mathsf{S}}, m^1_{\mathsf{S}}, a_0, a_1, u_0, u_1)$ to $\mathcal{A}$;
   - Sets $b \leftarrow \perp$. When $k^{\bar{b}}_{\mathsf{S}}$ is asked to the random oracle H_2, it sets $b \leftarrow \bar{b}$;
   - Aborts, if w_{1-b} is asked to the random oracle H_3 before w_b or if k^{1-b}_{S} is asked to H_2.
3. Upon receiving (sid, ch') from the adversary $\mathcal{A}(\mathsf{R})$, the simulator Sim:
   - Aborts, if $ch \neq ch'$;
   - If $b = \perp$, sets $b \leftarrow_{\$} \{0,1\}$;

- Sends (sid, b) to the ideal functionality $\mathcal{F}_{\mathrm{OT}}$.

4. Upon receiving (sid, M_b) from $\mathcal{F}_{\mathrm{OT}}$, the simulator Sim:
   - Encrypts $c_b \leftarrow \mathsf{Enc}_\Delta(k_{\mathsf{S}}^b, M_b)$ and $c_{1-b} \leftarrow \mathsf{Enc}_\Delta(k_{\mathsf{S}}^{1-b}, 0^\lambda)$;
   - Sends (sid, c_0, c_1) to $\mathcal{A}(\mathsf{R})$;
   - Halts whenever $\mathcal{A}(\mathsf{R})$ halts.

The executions differ when Sim aborts when was not supposed to. This happens if $\mathcal{A}$ asks the key k_{S}^{1-b} to the random oracle H_2, or if it asks w_{1-b} to the random oracle H_3 before w_b, or even if none of the keys k_{S}^0 and k_{S}^1 are queried to the random oracle. The first two cases have a negligible probability (in the security parameter) of happening. The last case also has negligible probability of happening since, without asking any of the keys, the adversary has negligible probability of guessing ch. It follows that

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathsf{Sim},\mathcal{E}} \stackrel{\mathrm{c}}{\approx} \mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi_{\mathrm{OT}},\mathcal{A}(\mathsf{R}),\mathcal{E}}. \qquad \square$$

Lemma 12. *Given any adversary $\mathcal{A}(\mathsf{S})$ corrupting the sender S, there is a simulator Sim such that for every environment $\mathcal{E}$ we have*

$$\mathsf{IDEAL}_{\mathcal{F}_{OT},\mathsf{Sim},\mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_{RO}}_{\pi_{OT},\mathcal{A}(\mathsf{S}),\mathcal{E}},$$

given that Δ is IND-CPA *secure and the A-B ORKE scheme used is ψ-message indistinguishable and ψ-key indistinguishable.*

Proof. In this case, the inputs of the sender are M_0 and M_1. The goal of the simulator is, given any adversary corrupting the sender, to extract the messages M_0 and M_1. The extraction is possible since the simulator can program the random oracle H_1, and this allows it to have both keys k_{S}^0 and k_{S}^1. Therefore, it is able to extract both messages from the ciphertexts c_0 and c_1.

Recall that, by assumption, given $x, y \in \overline{\mathcal{M}}$, it is computationally easy to find $h \in \mathbb{M}$ such that $x \leftarrow \psi(y, h)$. We specify how the simulator Sim proceeds.

1. Before activating the adversary, the simulator Sim:
   - Chooses $r_{\mathsf{R}}^0 \leftarrow\!\!\$\, \{0,1\}^\kappa$ and $r_{\mathsf{R}}^1 \leftarrow\!\!\$\, \{0,1\}^\kappa$;
   - Computes $m_{\mathsf{R}}^0 \leftarrow \mathsf{Msg}_\Pi(r_{\mathsf{R}}^0, \mathsf{sk}_{\mathsf{R}})$ and $m_{\mathsf{R}}^1 \leftarrow \mathsf{Msg}_\Pi(r_{\mathsf{R}}^1, \mathsf{sk}_{\mathsf{R}})$.
2. Upon activating the adversary, the simulator Sim sends $(\mathsf{sid}, t, m_{\mathsf{R}}^0)$:

- Simulates H_2, H_3 and H_4 in the following way: Sim keeps a list L_i for each H_i, for $i = 2, 3, 4$, which is initially empty. Whenever $\mathcal{A}(\mathsf{R})$ queries H_i on (sid, q), Sim checks if there is $(q, h) \in L_i$. If so, it returns h. Else, it chooses h uniformly at random, records the pair (q, h) in L_i and returns h.
- Simulates H_1 in the following way: when the adversary queries H_1 with (sid, t), the simulator answers h such that $m^1_{\mathsf{R}} = \psi(m^0_{\mathsf{R}}, h)$. For all other queries to H_1, it answers as the ideal functionality would.

3. Upon receiving $(\mathsf{sid}, m^0_{\mathsf{S}}, m^1_{\mathsf{S}}, a_0, a_1, u_0, u_1)$ from $\mathcal{A}$, the simulator Sim:
   - Computes the keys $k^0_{\mathsf{R}} \leftarrow \mathsf{Key}_{\Pi}(r^0_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, m^0_{\mathsf{S}})$ and $k^1_{\mathsf{R}} \leftarrow \mathsf{Key}_{\Pi}(r^1_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, m^1_{\mathsf{S}})$;
   - Proceeds as the honest receiver would do and computes ch';
   - Sends (sid, ch') to $\mathcal{A}$.
4. Upon receiving (sid, c_0, c_1) from $\mathcal{A}$, the simulator Sim:
   - Computes $M_0 \leftarrow \mathsf{Dec}_{\Delta}(k^0_{\mathsf{R}}, c_0)$ and $M_1 \leftarrow \mathsf{Dec}_{\Delta}(k^1_{\mathsf{R}}, c_1)$;
   - Sends (sid, M_0, M_1) to the ideal functionality $\mathcal{F}_{\mathrm{OT}}$.
5. Upon receiving $(\mathsf{sid}, \mathit{receipt})$ from $\mathcal{F}_{\mathrm{OT}}$, the simulator Sim halts whenever the adversary halts.

Note that the executions differ in the outputs given by the random oracle H_1. But the value h, returned by the simulator to the adversary, is computationally indistinguishable from uniformly chosen values since the A-B ORKE scheme used is message indistinguishable. Hence, the probability that the environment distinguishes both the real and the simulated execution of the random oracle H_1 is negligible. Hence,

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathsf{Sim},\mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi_{\mathrm{OT}},\mathcal{A}(\mathsf{S}),\mathcal{E}}. \qquad \square$$

Remark. In the case where only S is corrupted, one could think the simulator does not need to program H_1 to obtain the keys, since they are asked by the sender to H_2. However, a closer inspection reveals that this is not true, as the corrupted sender might just encrypt one a_i with a random key instead of k^i_{S}, and the simulator would not have the right key needed to extract the message M_i. By allowing the simulator to program the random oracle H_1, we are sure that the keys used to encrypt each a_i are the right ones. $\square$

5 Efficiency and comparison

Efficiency of the framework. Let κ be the security parameter. To ease the presentation, we assume that the SKE protocol Δ has keys of size κ and ciphertexts are of the same size as the plaintexts. Suppose that the messages being sent by the sender are of size λ. Let α be the size of the binary representation of elements of $\mathcal{M}$.

Although our scheme has four rounds, it has a low communication complexity since it only requires the exchange of $2\alpha + 2\lambda + 10\kappa$ bits of information, per iteration of the protocol. The first message by the sender carries $\alpha + \kappa$ bits of information, the second $2\alpha + 2\kappa + 6\kappa$ bits of information, the third message is just the answer to the challenge which is of size κ and, finally, the fourth message carries 2λ bits of information.

Our protocol is also very efficient in terms of computational complexity since it only requires to run twice the Gen_Π algorithm and the Msg_Π algorithm and three times the Key_Π algorithm. It requires 11 calls to the random oracle. All other operations (sum modulo 2 and concatenation of strings) are linear in the security parameter and should be quite fast to perform.

Comparison with other frameworks. The framework of [PVW08] requires the use of a dual-mode public-key encryption (PKE) scheme. However, very few dual-mode PKE are known. For example, finding a dual-mode RLWE PKE scheme is stated as an open problem in [LKHB17]. Their framework has just two rounds. However, since it relies on PKE schemes, a public key needs to be sent from the receiver to the sender. For post-quantum PKE schemes, this key can be too large, which makes the communication and the computational complexity rather cumbersome and the scheme impractical for real-life uses. Another bottleneck regarding the framework of [PVW08] is that it relies its security in the Common Reference String (CRS) model. In practice, the common reference string needs to be generated using a third party (which always raises security issues) or by some multiparty computation protocol, which is too inefficient.

The work of Barreto *et al.* [BDD+17] presents a framework for OT in the ROM, which can be instantiated using a PKE scheme with certain properties. One of these properties is that the space of public keys of the PKE scheme used must have a group structure for a certain operation. This property is too exclusive and immediately discards some of the most important post-quantum PKE schemes such as LWE [Reg05] or RLWE [LPR10] PKE schemes. Note that both the public keys of these schemes do not have a group structure for any operation (e.g., this set is not closed under addition). However, we think that this condition is too strong and, perhaps, it could be weaken to accept LWE and RLWE-based instances. The framework of [BDD+17] has three rounds. But again, a public key needs to be sent from the receiver to the sender which will be reflected in a high communication complexity. Besides that, the framework requires six encryptions and two decryptions, whereas ours just requires the exchange of a symmetric secret key.

6 Framework instantiations

In the following section we provide relevant cases of ORKE schemes that can be used to instantiate our framework. More concretely, we show that our framework can be used with Diffie-Hellman, Ding's KE and Supersingular Isogeny Diffie-Hellman.

6.1 DH-based OT

Consider the Diffie-Hellman (DH) KE protocol [DH76]. Let p be a prime and consider the group $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Let $g \in \mathbb{Z}_p$ be a generator of the multiplicative group $\mathbb{Z}_p^*$. We assume g to be a public parameter of the system (e.g. a standard one), known by all parties. The DH KE is defined by three algorithms:

- $\mathsf{Gen}_{\mathrm{DH}}(1^\kappa)$ outputs a secret key $x \in \mathbb{Z}_p^*$ and a public key $\mathsf{pk} \leftarrow g$.
- $\mathsf{Msg}_{\mathrm{DH}}(r_i, \mathsf{sk}_i)[= \mathsf{Msg}^{\mathsf{A}}_{\mathrm{DH}}(r_i, \mathsf{sk}_i) = \mathsf{Msg}^{\mathsf{B}}_{\mathrm{DH}}(r_j, \mathsf{sk}_j, \cdot)]$ which takes as input the secret x_i and generator g and outputs g^{x_i}.
- $\mathsf{Key}_{\mathrm{DH}}(r_i, \mathsf{sk}_i, m_j)$ which takes as input a message $m_j \leftarrow g^{x_j}$ and a secret key x_i and outputs $m_j^{x_i}$.

Note that DH KE is an ORKE scheme, which means that $\mathsf{Msg}_{\mathrm{DH}}$ is the same for both parties.

Recall that the Decisional Diffie-Hellman (DDH) assumption assumes that (g, g^x, g^y, g^{xy}) is computationally indistinguishable from (g, g^x, g^y, z) when $z \leftarrow\!\!\$\, \mathbb{Z}_p^*$.

Using the notation of Section 2.2, consider $\mathcal{M} = \overline{\mathcal{M}} = \mathbb{M} = \mathbb{Z}_p^*$, the operation $*$ to be the product modulo p and $\psi : \mathbb{Z}_p^* \times (\mathbb{Z}_p^*, *) \to \mathbb{Z}_p^*$ to be the action group defined as $\psi(y, h) = y * h \mod p$.

The properties of ψ-message indistinguishability and ψ-key indistinguishability follow directly from the hardness of DDH of base g in the group $\mathbb{Z}_p^*$. Consider the notation of Definition 7.

Lemma 13. *The DH KE protocol is ψ-message indistinguishable.*

Proof. Since g is a generator of $\mathbb{Z}_p^*$, the message sent by Alice to Bob is a random element from $\mathbb{Z}_p^*$ when it is computed using Msg_Π or using ψ. □ □

Lemma 14. *The DH KE protocol is ψ-key indistinguishable, given that the DDH assumption holds.*

Proof. Any key obtained using the $\mathsf{Key}_{\mathrm{DH}}$ algorithm should be of the form g^{xy}, where g^x is the output of the other party's $\mathsf{Msg}_{\mathrm{DH}}$, and y is the secret key of the party running this algorithm. As before, g^{xy} is a random element in $\mathbb{Z}_p^*$, and so indistinguishable from a uniform chosen values from $\mathbb{Z}_p^*$, given that the hardness of the DDH assumption holds. □ □

Therefore, we conclude that the DH KE can be used to instantiate the framework presented in this paper.

6.2 RLWE-based OT

The instantiation of this framework using Ding's KE was presented previously in [BDGM18] and this framework can be viewed as a generalization of their work. Here, we present a more generic instantiation using any RLWE-based KE scheme, such as [DXL12, Pei14, ADPS16].

Let $q > 2$ be a prime such that $q \equiv 1 \mod 2n$, $n \in N$ be a power of 2 and $R_q = \mathbb{Z}_q[x]/\langle(x^n + 1)\rangle$. Let χ_α be a discrete Gaussian distribution with parameter α.

Let $s \leftarrow_\$ R_q$. The RLWE assumption asks to distinguish $(a, as + e)$ where $e \leftarrow_\$ \chi_\alpha$ from (a, u) where $u \leftarrow_\$ R_q$ [LPR10]. The HNF-RLWE assumption is similar to the RLWE assumption, but $s \leftarrow_\$ \chi_\alpha$ [ACPS09].

Consider an RLWE-based KE scheme, which is secure given that the HNF-RLWE problem is hard. Let $(\mathsf{recMsg}, \mathsf{recKey})$ be any reconciliation mechanism, as the ones presented in [DXL12, Pei14], where recMsg receives as input a value $x_1 \in R_q$ and outputs the signal w of x_1 and a key K, and recKey receives as input a value $x_2 \in R_q$ and a signal w and it outputs a key K. Recall that a reconciliation mechanism is parameterized by a bound ξ_{rec} such that if x_1 and x_2 are close (meaning that $|x_1 - x_2| \leq \xi_{\mathsf{rec}}$), then

$$\Pr\left[K_1 = K_2 : (w, K_1) \leftarrow \mathsf{recMsg}(x_1), K_2 \leftarrow \mathsf{recKey}(x_2, w)\right] \geq 1 - \mathsf{negl}(\kappa)\,.$$

It is also required that, if x_1 is uniform, then K_1 is indistinguishable from a uniform value, even when given w, where $(w, K_1) \leftarrow \mathsf{recMsg}(x_1)$.

Let $a \leftarrow_\$ R_q$ be a public polynomial. The four algorithms that define any RLWE-based KE based are the following:

- $\mathsf{Gen}_{RLWE}(1^\kappa)$ chooses $s \leftarrow_\$ \chi_\alpha$ and outputs a secret key $\mathsf{sk} \leftarrow_\$ s$ and a public key $\mathsf{pk} \leftarrow as + 2e \mod q$ where $e \leftarrow_\$ \chi_\alpha$.
- $\mathsf{Msg}^{\mathsf{A}}_{RLWE}(r_{\mathsf{A}}, \mathsf{sk}_{\mathsf{A}})$ outputs the message $m_{\mathsf{A}} = \mathsf{pk}_{\mathsf{A}}$.
- $\mathsf{Msg}^{\mathsf{B}}_{RLWE}(r_{\mathsf{B}}, \mathsf{sk}_{\mathsf{B}}, m_{\mathsf{A}})$ computes $(w, K) \leftarrow \mathsf{recMsg}(m_A \mathsf{sk}_B + 2e')$, where $e' \leftarrow_\$ \chi_\alpha$, and outputs $m_{\mathsf{B}} = (\mathsf{pk}_{\mathsf{B}}, w)$.
- $\mathsf{Key}_{RLWE}(r_i, \mathsf{sk}_i, m_j)$ computes $k_i \leftarrow s_i \mathsf{pk}_j + 2e'_i$, where $e' \leftarrow_\$ \chi_\alpha$, and outputs the shared key $K \leftarrow \mathsf{recKey}(k_i, w)$.

RLWE-based KE schemes [DXL12, Pei14, ADPS16] are A-B ORKE scheme since Bob's message depends on Alice message.

Using the notation of Section 2.2, consider $\mathcal{M}$ to be the set of RLWE samples, that is, $\mathcal{M} = \{x : x = as + e \wedge s, e \leftarrow_\$ \chi_\alpha\}$, and $\overline{\mathcal{M}} = \mathbb{M} = R_q$, the operation $*$ to be the sum in R_q and $\psi : R_q \times (R_q, +) \rightarrow R_q$ to be the action group defined as $\psi(y, h) = y + h$.

Lemma 15. *RLWE-based KE is ψ-message indistinguishable given that the HNF-RLWE assumption holds.*

Proof. The message algorithm of Alice ($\mathsf{Msg}^{\mathsf{A}}_{RLWE}$) in this key exchange protocol outputs messages which are HNF-RLWE samples, thus, it is trivial to reduce the problem of breaking ψ-message indistinguishability of an RLWE-based KE to the problem of deciding the HNF-RLWE problem. □ □

For the ψ-key indistinguishability property, let K_A and K_B be the output of the algorithm Key_{DingKE} when run by party A and B respectively.

Lemma 16. *RLWE-based KE protocol is ψ-key indistinguishable, given that the HNF-RLWE assumption holds.*

Proof. This follows directly from the security of the KE protocol. As proved in [DXL12, Theorem 3], to computationally distinguish K_A or K_B from uniformly random in R_q reduces to the HNF-RLWE assumption. Thus, if the HNF-RLWE assumption holds, the protocol is ψ-key indistinguishable. □ □

We conclude that RLWE-based KE schemes [DXL12, Pei14, ADPS16] can be used to instantiate the framework of this article.

6.3 SIDH-based OT

Following the work of [BOB18], where it is presented an OT protocol based on the Supersingular Isogeny Diffie-Hellman (SIDH) of [JDF11], we adapt the same techniques to achieve the first UC OT based on Supersingular Isogeny cryptography. Although we use the same techniques to instantiate our framework using this key exchange, we work in the ROM instead of using the secure coin flip they use.

As defined in [JDF11], let $p = \ell_A^{e_A}\ell_B^{e_B} \cdot f \pm 1$ where ℓ_A, ℓ_B are small primes and f is a cofactor such that p is prime. Let E_0 be a supersingular curve defined over $\mathbb{F}_{p^2}$, and let P_A, Q_A be a basis generating $E_0[\ell_A^{e_A}]$ and P_B, Q_B a basis generating $E_0[\ell_B^{e_B}]$, where $E[\ell]$ is the ℓ-torsion group of E, i.e. the set of all points $P \in E(\overline{\mathbb{F}}_q)$ such that ℓP is the identity. As in [BOB18], we consider $(P_A, Q_A), (P_B, Q_B)$ as public parameters of the cryptosystem.

Like the DH scheme, this is a vanilla ORKE scheme, since Msg_{SIDH} is the same for both parties, and does not depend on the message previously exchanged by the other party. The three algorithms that define the KE are:

- $\mathsf{Gen}_{SIDH}(1^\kappa, r)$ pick $m_i, n_i \in \mathbb{Z}/\ell_i^{e_i}\mathbb{Z}$, where at most one of them is divisible by ℓ_i, and compute an isogeny $\phi_i : E_0 \to E_i$ with kernel $K_i = \langle [m_i]P_i + [n_i]Q_i \rangle$. Set $\mathsf{sk} \leftarrow (m_i, n_i, \phi_i)$.
- $\mathsf{Msg}_{SIDH}(r_i, \mathsf{sk}_i)[= \mathsf{Msg}^{\mathsf{A}}_{SIDH}(r_A, \mathsf{sk}_A) = \mathsf{Msg}^{\mathsf{B}}_{SIDH}(r_B, \mathsf{sk}_B, \cdot)]$ compute images
$$\{\phi_i(P_j), \phi_i(Q_j)\} \subset E_i$$
and outputs the message $m = (E_i, \phi_i(P_j), \phi_i(Q_j))$.

- $\mathsf{Key}_{SIDH}(r_i, \mathsf{sk}_i, m_j)$ since $m_j \leftarrow (E_j, \phi_j(P_i), \phi_j(Q_i))$, compute an isogeny $\phi'_i : E_j \to E_{ij}$ considering its kernel $\langle [m_i]\phi_j(P_i) + [n_i]\phi_j(Q_i)\rangle$. Return the *j-invariant* of

$$\begin{aligned} E_{AB} &= \phi'_A(\phi_B(E_0)) = \phi'_B(\phi_A(E_0)) \\ &= {}^{E_0}\big/_{\langle [m_A]P_A + [n_A]Q_A, [m_B]P_A + [n_B]Q_B\rangle}. \end{aligned}$$

Now, we prove that there exists the group action ψ as stated in Definition 7. Again, we base our group action on the assumptions of [BOB18] and follow their notation. Consider $\mathcal{M} = \overline{\mathcal{M}}$ to be the set of elements of the form (E, G, H), where G and H are elements of the ℓ-torsion group of E. In [BOB18], it is assumed that (E, G, H) is computationally indistinguishable from $(E, G+U, H+V)$ when U, V are randomly chosen among $E[\ell]$ such that the Weil paring of (G, H) and $(G + U, H + V)$ coincides. Moreover, they also show that such U, V can be sampled in polynomial time among the elements of $E[\ell]$, namely $U \leftarrow \alpha G_B + \beta H_B$, $V \leftarrow -(\alpha/\beta)U$, where $G_B \leftarrow \phi_B(P_A)$, $H_B \leftarrow \phi_B(Q_A)$, and $\alpha, \beta \in \mathbb{Z}/\ell\mathbb{Z}$.

We are now able to propose the required group action ψ. Let $\mathbb{M}$ be the group of elements of the form $(U, V) \in E[\ell]$ with group law $*$ being the coordinate-wise usual sum of the ellipic curve points. This group acts on $\overline{\mathcal{M}}$, $\psi : \overline{\mathcal{M}} \times (\mathbb{M}, *) \to \overline{\mathcal{M}}$, by modifying G and H, as $\psi(y, h) = (E, G + U, H + V)$, where y is of the form of (E, G, H) and h of the form (U, V), and G, H, U, V are all elements in $E[\ell]$, such that U, V are sampled accordingly with [BOB18].

Lemma 17. *The SIDH KE protocol is ψ-message indistinguishable given the security assumptions in [JDF11, Section 5] and the parameters are chosen as to prevent any distinguisher based attack [BOB18].*

Proof. In order to achieve the property of ψ-message indistinguishability, we must prevent any distinguisher from figuring out if the first message from the receiver is (E, G, H) or $(E, G + U, H + V)$. As in [BOB18], we can choose the parameters to avoid the paring-based distinguisher using the Weil pairing, and so prevent the sender from finding out the secret bit of the receiver. If their conjecture that there is no other polynomial-time distinguisher for schemes of this form holds, then our OT protocol is ψ-message indistinguishable. □ □

Note that, differently from [BOB18], in our proposal the receiver sends either (E, G, H) or $(E, G + U, H + V)$, together with the nounce t such that $(U, V) \leftarrow \mathsf{H}(t)$. In fact, [BOB18] uses a secure coin flip procedure to generate U, V, while in this work we obtain U, V from the random oracle. This means that the receiver has the ability to try a polynomial number of queries to the RO in order to choose U, V, in contrast to the single possibility of [BOB18]. Notwithstanding, if it would be possible for the receiver to obtain a *good* U, V in polynomial many tries, then the probability of the secure coin flip would be non-negligible. Therefore, the two approaches are equivalent with regard to the security of this procedure.

Lemma 18. *The SIDH KE is ψ-key indistinguishable given the assumptions in [JDF11, Section 5].*

Proof. This follows from the proof of security of the key exchange in [JDF11]. The shared key must be a *j-invariant* uniformly random in the set *j-invariants*, i.e. a random curve in the isogeny graph, which according to the assumptions in [JDF11, Section 5] is difficult to compute without knowledge of the private isogenies. □ □

Therefore, we conclude that SIDH KE protocol of [JDF11] can be used to instantiate the framework in this article.

Acknowledgment

The first author thanks the support from DP-PMI and FCT (Portugal) through the grant PD/BD/ 135181/2017. This work was done while visiting the University of Cincinnati. The third author thanks the support from DP-PMI and FCT (Portugal) through the grand PD/BD/135182/2017.

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, *Advances in Cryptology - CRYPTO 2009*, pages 595–618, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.

[ALSZ17] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions. *J. Cryptol.*, 30(3):805–858, 2017.

[BDD+17] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. Cryptology ePrint Archive, Report 2017/993, 2017. `https://eprint.iacr.org/2017/993`.

[BDGM18] Pedro Branco, Jintai Ding, Manuel Goulão, and Paulo Mateus. Universally composable oblivious transfer protocol based on the RLWE assumption. Cryptology ePrint Archive, Report 2018/1155, 2018. `https://eprint.iacr.org/2018/1155`.

[BJS15] Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 477–494, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BM90] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 547–557, New York, NY, 1990. Springer New York.

[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`.

[BPRS17] Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. `https://eprint.iacr.org/2017/1165`.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 19–40, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 453–474, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 494–503, New York, NY, USA, 2002. ACM.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology – LATINCRYPT 2015*, pages 40–58, Cham, 2015. Springer International Publishing.

[DDN14] Bernardo M. David, Rafael Dowsley, and Anderson C. A. Nascimento. Universally composable oblivious transfer based on a variant of LPN. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis

Askoxylakis, editors, *Cryptology and Network Security*, pages 143–158, Cham, 2014. Springer International Publishing.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Nov 1976.

[DKLas18] J. Doerner, Y. Kondi, E. Lee, and a. shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 595–612, 2018.

[DNMQ12] Bernardo M. David, Anderson C. A. Nascimento, and Jörn Müller-Quade. Universally composable oblivious transfer from lossy encryption and the McEliece assumptions. In Adam Smith, editor, *Information Theoretic Security*, pages 80–99, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. `https://eprint.iacr.org/2012/688`.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography*, pages 303–324, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[HL17] Eduard Hauck and Julian Loss. Efficient and universally composable protocols for oblivious transfer from the CDH assumption. Cryptology ePrint Archive, Report 2017/1011, 2017. `https://eprint.iacr.org/2017/1011`.

[JDF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[JKL04] Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-round protocols for two-party authenticated key exchange. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security*, pages 220–232, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM.

[KO97] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Oct 1997.

[LKHB17] Mo-meng Liu, Juliane Krämer, Yu-pu Hu, and Johannes Buchmann. Quantum security analysis of a lattice-based oblivious transfer protocol. *Frontiers of Information Technology & Electronic Engineering*, 18(9):1348–1369, Sep 2017.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography*, pages 197–219, Cham, 2014. Springer International Publishing.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 554–571, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Rab81] Michael O Rabin. How to exchange secrets with oblivious transfer. 1981.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.