Universally Composable Oblivious Transfer Protocol based on the RLWE Assumption

Pedro Branco*[1,2], Jintai Ding[3], Manuel Goulão[1,2], and Paulo Mateus[1,2]

[1]SQIG-IT
[2]Department of Mathematics, IST-Universidade de Lisboa
[3]Department of Mathematical Sciences, University of Cincinnati

November 26, 2018

Abstract

We use an RLWE-based key exchange scheme to construct a simple and efficient post-quantum oblivious transfer based on the Ring Learning with Errors assumption. We prove that our protocol is secure in the Universal Composability framework against static malicious adversaries in the random oracle model. The main idea of the protocol is that the receiver and the sender interact using the RLWE-based key exchange in such a way that the sender computes two keys, one of them shared with the receiver. It is infeasible for the sender to know which is the shared key and for the receiver to get information about the other one. The sender encrypts each message with each key using a symmetric-key encryption scheme and the receiver can only decrypt one of the ciphertexts. The protocol is extremely efficient in terms of computational and communication complexity, and thus a strong candidate for post-quantum applications.

1 Introduction

Oblivious transfer (OT) [Rab81] is a cryptographic primitive involving two parties, usually called the sender S and the receiver R. It allows S to send two messages to R in such a way that: i) R chooses one message to receive and gets no information on the other one, and ii) S does not know which message the other party has received. Despite its simplicity, OT is a powerful tool since it can be used to create other more complex protocols. Undoubtedly, the most important application of OT is its use in the construction of (almost) every efficient and secure multi-party computation protocol (e.g., Yao's protocol [Yao86]).

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

Since OT is mostly used as a building block inside other more sophisticated protocols, its security should be analyzed in the Universal Composability (UC) framework of Canetti [Can01]. Using this framework, security under arbitrary composition can be guaranteed.

In this work, we present a practical UC-secure OT protocol. Since its security is based on the RLWE assumption, the protocol is conjectured to be robust to quantum adversaries.

Previous work. In recent years, several proposals for UC-secure OT protocols have been made [PVW08, DDN14, BDD$^+$17, HL17, BPRS17, DKLas18], being the most efficient ones based on number-theoretic assumptions [PVW08, BDD$^+$17, HL17, DKLas18]. Unfortunately, Shor's algorithm makes them useless in case a quantum computer ever exists [Sho97]. Hence, the development of efficient post-quantum OT protocols is crucial if we want to securely implement oblivious transfer in the post-quantum era. The post-quantum UC-secure proposals for OT include a McEliece-based OT protocol [DNMQ12], a LPN-based OT protocol [DDN14] and the ones that can be obtained from instantiating the frameworks of [PVW08, BDD$^+$17] using post-quantum cryptosystems. However, these proposals require the use of public-key encryption schemes and this leads to extremely high communication costs since public keys need to be sent from one party to the other, at some point in the protocol. Another approach, usually more efficient in terms of communication and computational complexity, is to use a key exchange scheme and encrypt with a symmetric-key encryption scheme. OT proposals based on this paradigm were presented in [CO15, BOB18], although none of them is UC-secure (the protocol in [CO15] was originally claimed to be universally composable but, later, this turned out to be false [BDD$^+$17, BPRS17, DKLas18]).

The Ring Learning with Errors (RLWE) problem [LPR10] is a variant of the LWE problem [Reg05], which allows to construct more efficient and compact cryptosystems. It is conjectured that this problem is hard, even for quantum adversaries. Therefore, cryptographic primitives based on this problem are usually very efficient and conjectured to be post-quantum secure. However, it was still an open problem to construct UC-secure OT protocols based on the RLWE problem, until this moment. Recall that the framework of [PVW08] requires a dual-mode encryption scheme and it can be instantiated using the LWE public-key encryption scheme. It is not known if there exists a dual-mode variant of the RLWE public-key encryption scheme, a problem stated as open in [LKHB17]. Thus, it is not known if it is possible to use the framework of [PVW08] to construct a RLWE-based OT protocol. Moreover, the framework of [BDD$^+$17] requires a group structure in the set of the public keys of the cryptosystem to be used. For this reason, it is not possible to use the LWE nor the RLWE public-key encryption schemes in the framework of [BDD$^+$17] since the set of the public keys of these schemes do not have a group structure for any operation.

After our work was essentially finished, there appeared a paper [LH18] where a proposal for OT based on the RLWE assumption and achieving security in the UC-framework was presented. However, the scheme does not provide security for the receiver. More precisely, the signal function (used in the reconciliation mechanism of the RLWE-based key exchange) is sent from the receiver R to the sender S. This allows S to figure out R's input, since the signal function matches the right input, and this is very much related to the fact that the signal function leaks information [DRAC18]. Also, the same argument in [BDD$^+$17, BPRS17, DKLas18] that invalidates the proof in the UC-framework of the simplest OT can be applied to the scheme [LH18]. Hence, the claim that the protocol achieves security in the UC-framework is false. More details on this are presented below.

Contribution of this work. The RLWE-based key exchange by Ding *et al.* [DXL12] has proven to be very versatile as it is the core of several lattice-based constructions (e.g. [ZZD$^+$15, DAL$^+$17]). In this work, we use the RLWE-based key exchange protocol to build a post-quantum universally composable $\binom{2}{1}$-OT, in the random oracle model. We also show how the protocol can be easily extended to obtain a $\binom{N}{1}$-OT protocol. Our protocol can be seen as a adaptation of the simplest OT protocol [CO15] using the RLWE key exchange scheme instead of Diffie-Hellman key exchange scheme [DH76]. In a nutshell, the receiver and the sender interact using the key exchange protocol and, at the end, the sender has two keys, one of them shared with the receiver. The sender encrypts each message with each key, and the receiver will only be able to decrypt one of them.

Contrarily to [CO15, LH18], where the interaction begins with S, in our protocol the interaction begins with R. The reasons for this are: i) it allows to reduce the number of rounds of the protocol, and ii) it avoids the attack of [DRAC18] since, in this case, the signal function is sent by S and not by R. As mentioned before, if the signal function is sent from R to S, then S has enough information to extract the input of R with non-negligible probability. Informally, in [CO15, LH18], S plays the role of Alice in the key exchange, while in our protocol it is R that plays the role of Alice.

In the simplest OT protocol, R uses as key an output of the random oracle. To argue security in the UC-framework against a corrupted R, the simulator uses R's queries to the random oracle to extract its input. However, as noted in [BDD$^+$17, BPRS17, DKLas18], the corrupted receiver can just delay the query of the key to the random oracle and the decryption and the simulator will have no information to perform the simulation. In this case, S in the real-world execution halts while the simulator in the ideal-world is still running and, thus, an environment could distinguish both executions. To achieve security in the UC-framework for our protocol, we have to somehow force R to query the random oracle with the key. This idea was already explored in [BDD$^+$17], where S makes a challenge to R, that it can only answer correctly if it queries its key to the random oracle. However, the scheme of [BDD$^+$17] uses a public-key encryption scheme, while ours uses a key exchange followed by a symmetric-key encryption scheme. Therefore, the adaptation of this technique is not straightforward, since, in our case, every key in the protocol is secret. We solve this problem by allowing R to know the output of both secret keys by the random oracle, and using these outputs as keys in the challenge. Note that this does not affect the security of the scheme, since the random oracle should output a completely random value, and, thus, it should be infeasible for R to know their pre-image.

Although the scheme proposed in this paper has four rounds, after analyzing its efficiency, we conclude that it improves state-of-the-art post-quantum OT protocols in terms of computational and communication complexity. Let $q \in \mathbb{Z}$, $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ and κ be the security parameter of the OT protocol. Our scheme achieves a communication complexity of $\mathcal{O}(\lambda \log q + \kappa)$ and it just requires five multiplications in the ring R_q and six samples from the RLWE error distribution to obliviously send a λ-bit message. Whereas the frameworks of [PVW08, BDD$^+$17] require a public key of a post-quantum public-key encryption scheme to be generated and transmitted. Recall that the size of public keys of post-quantum public-key encryption schemes is one of the main drawbacks of post-quantum cryptography. Hence, the use of post-quantum public-key encryption schemes in OT leads to high communication and computational complexity since their public keys are, generally, too large for practical purposes. More precisely, the framework of [PVW08] using the LWE public-key encryption scheme has communication and computational complexity of $\mathcal{O}(\lambda^3)$ to obliviously send a λ-bit message, and the framework of [BDD$^+$17], when instantiated with (for example) McEliece public-key encryption scheme, achieves complexity $\mathcal{O}(\lambda^2)$. Therefore, as far as we know, our protocol is the most efficient post-quantum OT protocol presented so far. In real-life application, using hardware implementations of AES and SHA-3, we believe that our protocol should be very practical and it should be considered when implementing post-quantum secure multi-party computation.

Roadmap. In Section 2 we state relevant concepts and definitions. Then we present our universally composable OT protocol based on the RLWE problem in Section 3. Section 4 shows the details of the proof of security in the UC framework. Finally, we analyze the communication and computation complexity of the protocol in Section 5.

2 Preliminaries

As usual, $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{Z}$ denotes the set of integers, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, for any $q \in \mathbb{N}$, and $\mathbb{Z}[X]$ (resp. $\mathbb{Z}_q[X]$) denotes the ring of polynomials on variable X with coefficients in $\mathbb{Z}$ (resp. $\mathbb{Z}_q$). 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, we denote by $x \leftarrow_{\$} S$ the experiment of choosing uniformly at random an element x from S. If χ is a probabilistic distribution over some set S, $x \leftarrow_{\$} \chi$ denotes the experiment of sampling an element 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.

We present the definition of a symmetric-key encryption scheme which will be an important tool to design the new OT protocol.

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

- $c \leftarrow \mathsf{SEnc}_\Delta(k, M; r)$ is a PPT algorithm that takes as input a shared key k, a message to encrypt M and some randomness r and outputs the ciphertext c. Whenever r is omitted, it means that it was chosen uniformly at random;

- $M/\perp \leftarrow \mathsf{SDec}_\Delta(k, c)$ is a PPT algorithm that takes as input a key k and a ciphertext c and outputs either a message M, if c was encrypted using k, or an error message $\perp$, otherwise.

A symmetric-key encryption scheme must be sound, that is, $M \leftarrow \mathsf{SDec}_\Delta\,(k, \mathsf{SEnc}_\Delta\,(k, M; r))$ for any message M and any r. It also should be secure, that is, it is infeasible for any PPT adversary to recover M from a ciphertext c without knowing the secret key k.

2.1 UC-security and ideal functionalities

The Universal Composability (UC) framework, introduced by Canetti [Can01], ensures that the security of a protocol does not depend on other executions of the same or other protocols. In a nutshell, given any protocol π, we say that it is UC-secure if no environment $\mathcal{E}$ (an entity that oversees both executions) can distinguish between the execution of π in the real-world and the execution of an ideal functionality $\mathcal{F}$ (defined *a priori*) in the ideal-world. By proving that both executions are indistinguishable from the point-of-view of the environment $\mathcal{E}$, we show that any cheating strategy that an adversary uses in the real-world execution of the protocol, it could also be used in the ideal-world execution by a simulator. Since the ideal functionality $\mathcal{F}$ is defined in such a way that each party involved learns nothing more than its own input and output, we conclude that it is infeasible to extract more information than this one.

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.

Using random oracles in the UC framework. Since UC-security is a very strong level of security, it is not odd if some cryptographic primitives turned out to be impossible to achieve within this framework. That is what happens with oblivious transfer or bit commitment, for example [CF01]. For this reason, we work on the so called $\mathcal{F}_{\mathrm{RO}}$-hybrid model in order to model random oracles in the UC framework. The random oracle ideal functionality $\mathcal{F}_{\mathrm{RO}}$ is presented below. Let $\mathcal{D}$ be the range of the random oracle and L be a list, which is initially empty. The value sid represents the session ID and the parties involved in the protocol.

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

Upon receiving a query $(\mathsf{sid}|q)$ from a party 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 $(\mathsf{sid}|h)$;
- Else, it chooses $h \leftarrow_{\$} \mathcal{D}$, stores the pair $(q, h) \in L$ and returns $(\mathsf{sid}|h)$.

In the $\mathcal{F}_{\mathrm{RO}}$-hybrid model, every party involved in the real-world execution of the protocol (and the adversary) have access to an auxiliary ideal functionality $\mathcal{F}_{\mathrm{RO}}$. The environment has access to this ideal functionality through the adversary. Similarly, 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 adversary $\mathcal{A}$, in which every party involved in the protocol have access to the ideal functionality $\mathcal{F}_{\mathrm{RO}}$, apart from the environment which has access through the adversary. An analogous definition of UC-security can be made in the $\mathcal{F}_{\mathrm{RO}}$-hybrid model. More precisely, we say that a protocol π UC-realizes $\mathcal{F}$ in the $\mathcal{F}_{\mathrm{RO}}$-hybrid model if for every PPT adversary $\mathcal{A}$ there is a PPT simulator Sim such that for all PPT environments $\mathcal{E}$,

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

Adversarial model. The adversarial model we adopt in this work is the static (or non-adaptive) corruption adversarial model [Can01]. In this model, adversaries may deviate in any way from the protocol but the corruption of each party happens before the beginning of the protocol.

Oblivious transfer. Oblivious transfer (OT) [Rab81] is a crucial primitive in cryptography since it is used as a building block to construct other primitives, such as bit commitment or secure multiparty computation. The $\binom{2}{1}$-OT ideal functionality $\mathcal{F}_{\text{OT}}$ is presented below, as in [CLOS02]. The functionality involves two parties, the sender S and the receiver R. Let $\lambda \in \mathbb{N}$ be a fixed value known to both parties, $M_0, M_1 \in \{0,1\}^\lambda$ and $b \in \{0,1\}$.

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

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

2.2 RLWE problem and RLWE key exchange

The RLWE problem [LPR10] is the ring version of the LWE problem [Reg05] and it is conjectured to be hard for both classical and quantum computers. Before presenting the problem, we define the RLWE distribution. Let $n = 2^a$ for some $a \in \mathbb{N}$, $q \geq 2$, $R_q = \mathbb{Z}_q[X]/\langle f(X)\rangle$ where $f(X) = X^n + 1$ and χ be the error distribution (which is usually a discrete Gaussian [LPR10]) and which satisfies $\Pr[\|p\| > \beta : p \leftarrow_{\$} \chi] \leq \mathsf{negl}(n)$ for some $\beta \in \mathbb{N}$, where $\|p\| = \|p\|_\infty = \max\{p_i\}_i$ denotes the largest coefficient of the polynomial $p = p_0 + p_1 X + \dots p_{n-1}X^{n-1} \in R_q$. For $s \in R_q$, the RLWE distribution $A_{s,\chi}$ is obtained by choosing $a \leftarrow_{\$} R_q$, $e \leftarrow_{\$} \chi$ and outputting $(a, as + e \mod q)$.

Problem 3 (Ring Learning with Errors)**.** *Let n, q, R_q, χ and $A_{s,\chi}$ be as above. The decision version of the* RLWE *problem is the following: for $s \leftarrow_{\$} R_q$, distinguish the case when it is given a polynomial number of samples from $A_{s,\chi}$ or when it is given uniformly chosen at random values.*

The problem is proven to be as hard as quantumly solving a worst-case lattice problem (the approximate SVP) which is conjectured to be hard for both classical and quantum computers [LPR10]. The advantages of using the RLWE assumption instead of the LWE assumption is that the keys of the primitives based on RLWE are smaller than the ones based on LWE, and, since the RLWE uses polynomials in the ring R_q we can use the Fast Fourier Transform (FFT) to enhance the speed of the multiplication. Here, we use the Hermite Normal Form of the RLWE problem, usually called HNF-RLWE, in which the secret s is sampled from the error distribution χ instead of being chosen uniformly at random from the ring R_q. This version of the problem is also assumed to be hard [ACPS09].

We define the signal function Sig and the extraction function Mod_2 as in [DXL12]. Both of these functions are used in the reconciliation mechanism of the key exchange protocol in [DXL12] and allows the parties involved to compute a shared key. Let $\sigma_0, \sigma_1 : \mathbb{Z}_q \to \{0,1\}$ such that

$$\sigma_0(a) = \begin{cases} 0, & a \in \left[-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor\right] \\ 1, & \text{otherwise} \end{cases}$$

and

$$\sigma_1(a) = \begin{cases} 0, & a \in \left[-\lfloor \frac{q}{4} + 1 \rfloor, \lfloor \frac{q}{4} + 1 \rfloor\right] \\ 1, & \text{otherwise} \end{cases}$$

for $a \in \mathbb{Z}_q$. When $a = \sum_{i=0}^{n-1} a_i X^i \in R_q$, then $\sigma_0(a) = \sum_{i=0}^{n-1} \sigma_0(a_i) X^i$ and $\sigma_1(a) = \sum_{i=0}^{n-1} \sigma_1(a_i) X^i$. The signal function $\mathsf{Sig} : R_q \to R_2$ is defined as $\mathsf{Sig}(a) = \sigma_b(a)$ where $b \leftarrow_{\$} \{0,1\}$. The extraction function $\mathsf{Mod}_2 : R_q \times R_2 \to R_2$ is defined as

$$\mathsf{Mod}_2(a, \sigma) = \left(a + \sigma \frac{q-1}{2} \mod q\right) \mod 2.$$

We now present Ding's key exchange (KE) protocol based on RLWE. Let q, n, χ, R_q be as above and $m \leftarrow_{\$} R_q$.

RLWE-based key exchange

Alice		**Bob**
$s_A, e_A, e'_A \leftarrow_{\$} \chi$		
$p_A \leftarrow m s_A + 2e_A \mod q$	$\xrightarrow{p_A}$	$s_B, e_B, e'_B \leftarrow_{\$} \chi$
		$p_B \leftarrow m s_B + 2e_B \mod q$
	$\xleftarrow{(p_B, \sigma)}$	$k_B \leftarrow p_A s_B + 2e'_B \mod q$
$k_A \leftarrow s_A p_B + 2e'_A \mod q$		$\sigma \leftarrow \mathsf{Sig}(k_B)$
$\mathsf{sk} \leftarrow \mathsf{Mod}_2(k_A, \sigma)$		$\mathsf{sk} \leftarrow \mathsf{Mod}_2(k_B, \sigma)$

The soundness of the scheme as well as it security are proven in [DXL12]. The protocol has its security based on the HNF-RLWE problem since the secret is sampled from the error distribution.

Regarding its efficiency, the scheme requires $2\mu \log q + \mu$ bits of information to be exchanged, where μ is the length of the key exchanged, and four multiplications in the ring R_q and six samplings from χ [DXL12].

3 An universally composable OT based on the RLWE problem

Let κ be the security parameter, $\Delta = (\mathsf{SEnc}_\Delta, \mathsf{SDec}_\Delta)$ be a symmetric-key encryption protocol and n, q, R_q, χ as described in the previous section. Let

$m \leftarrow_\$ R_q$ be a polynomial common to every party in the protocol. In our construction we need four instances of the random oracle functionality $\mathcal{F}_{\mathrm{RO}}$: H_1 with range R_q, H_2 with range $\mathcal{K} = \{0,1\}^\beta$ where β is the size of the keys outputted by the RLWE-based key exchange, H_3 with range $\{0,1\}^{2\kappa+\beta}$ and H_4 with range $\{0,1\}^\kappa$.

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 the input of R. Let sid denote the session ID.

The protocol. The new OT protocol has 4 rounds. We will call it π_{OT}.

- R starts by sampling $s_\mathsf{R}, e_\mathsf{R}, e'_\mathsf{R} \leftarrow_\$ \chi$ and computing $p^b_\mathsf{R} \leftarrow ms_\mathsf{R} + 2e_\mathsf{R} \mod q$. It also chooses $r \leftarrow_\$ \{0,1\}^\kappa$. It queries the random oracle H_1 with $(\mathsf{sid}|r)$, setting the output to h. If $b=1$, it computes $p^0_\mathsf{R} = p^1_\mathsf{R} - h \mod q$. When $b = 0$, the receiver does not have to compute p^1_R since, in this case, it will never use this value. It sends sid, p^0_R and r to S.
- Upon receiving sid, p^0_R and r, S samples $s_\mathsf{S}, e_\mathsf{S}, e'_\mathsf{S} \leftarrow_\$ \chi$ and computes $p_\mathsf{S} \leftarrow ms_\mathsf{S} + 2e_\mathsf{S} \mod q$. It queries H_1 with $(\mathsf{sid}|r)$ to obtain h' and sets $p^1_\mathsf{R} = p^0_\mathsf{R} + h' \mod q$. Now, it can compute the values

  $$k^0_\mathsf{S} \leftarrow s_\mathsf{S} p^0_\mathsf{R} + 2e'_\mathsf{S} \mod q \quad \text{and} \quad k^1_\mathsf{S} \leftarrow s_\mathsf{S} p^1_\mathsf{R} + 2e'_\mathsf{S} \mod q,$$

  and the respective signal of these values, that is,

  $$\sigma_0 \leftarrow \mathsf{Sig}(k^0_\mathsf{S}) \quad \text{and} \quad \sigma_1 \leftarrow \mathsf{Sig}(k^1_\mathsf{S}).$$

  It computes the keys

  $$\mathsf{sk}^0_\mathsf{S} \leftarrow \mathsf{Mod}_2(k^0_\mathsf{S}, \sigma_0) \quad \text{and} \quad \mathsf{sk}^1_\mathsf{S} \leftarrow \mathsf{Mod}_2(k^1_\mathsf{S}, \sigma_1).$$

  Now, S will make a challenge to R. It chooses $w_0, z_0, w_1, z_1 \leftarrow \{0,1\}^\kappa$. It queries H_2 with $(\mathsf{sid}|\mathsf{sk}_0)$ and $(\mathsf{sid}|\mathsf{sk}_1)$, setting the outputs to $\bar{\mathsf{sk}}^0_\mathsf{S}$ and to $\bar{\mathsf{sk}}^1_\mathsf{S}$, and queries H_3 with $(\mathsf{sid}|w_0)$ and $(\mathsf{sid}|w_1)$, setting the output to $\bar{w}_0$ and $\bar{w}_1$, respectively. It computes

  $$a_0 \leftarrow \mathsf{SEnc}_\Delta(\bar{\mathsf{sk}}^0_\mathsf{S}, w_0; z_0) \quad \text{and} \quad a_1 \leftarrow \mathsf{SEnc}_\Delta(\bar{\mathsf{sk}}^1_\mathsf{S}, w_1; z_1)$$

  and sets

  $$u_0 = \bar{w}_0 \oplus (w_1|\bar{\mathsf{sk}}^1_\mathsf{S}|z_1) \quad \text{and} \quad u_1 = \bar{w}_1 \oplus (w_0|\bar{\mathsf{sk}}^0_\mathsf{S}|z_0).$$

  Finally, it asks H_4 for $(\mathsf{sid}|w_0|w_1|z_0|z_1)$ and sets the result to a challenge ch that it keeps to itself. It sends sid, p_S, σ_0, σ_1, a_0, a_1, u_0 and u_1 to R.
- After receiving these values, the goal of R is to compute ch. First, it obtains the secret shared key by computing $k_\mathsf{R} \leftarrow p_\mathsf{S} s_\mathsf{R} + 2e'_\mathsf{R} \mod q$ and then $\mathsf{sk}_\mathsf{R} \leftarrow \mathsf{Mod}_2(k_\mathsf{R}, \sigma_b)$. Then, it decrypts $x_b \leftarrow \mathsf{SDec}_\Delta(\bar{\mathsf{sk}}_\mathsf{R}, a_b)$

where $\bar{\mathsf{sk}}_{\mathsf{R}}$ is the value returned by H_2 on input $(\mathsf{sid}|\mathsf{sk}_{\mathsf{R}})$. It asks H_3 for $(\mathsf{sid}|x_b)$ to get $\bar{x}_b$ and then it can recover $(x_{1-b}|\bar{\mathsf{sk}}_{\mathsf{R}}^{1-b}|y_{1-b}) = u_b \oplus \bar{x}_b$. To recover y_b, it asks H_3 for $(\mathsf{sid}|x_{1-b})$ to get $\bar{x}_{1-b}$ and then it recovers $(x'_b|\bar{\mathsf{sk}}_{\mathsf{R}}^{b}|y_b) = u_{1-b} \oplus \bar{x}_{1-b}$. Having recovered x_0, y_0, x_1 and y_1, it can query H_4 on input $(\mathsf{sid}|x_0|x_1|y_0|y_1)$ and sets the output to ch'. To be sure that S has not cheated, R verifies if $a'_0 = a_0$, if $a'_1 = a_1$ where $a'_0 \leftarrow \mathsf{SEnc}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{R}}^{0}, x_0; y_0)$ and $a'_1 \leftarrow \mathsf{SEnc}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{R}}^{1}, x_1; y_1)$, if $\bar{\mathsf{sk}}_{\mathsf{R}}^{b} = \bar{\mathsf{sk}}_{\mathsf{R}}$ and if $x'_b = x_b$. It aborts if any of these conditions fail. Else, it sends sid and ch' to S.

- After receiving ch' from the R, the sender checks if $ch = ch'$. If the test fails, it aborts the protocol. Otherwise, it encrypts both messages M_0 and M_1 using SEnc_{Δ} with keys $\mathsf{sk}_{\mathsf{S}}^0$ and $\mathsf{sk}_{\mathsf{S}}^1$, respectively, and sends the resulting ciphertexts c_0 and c_1 and the session ID sid to the receiver. It halts.

  Finally, after receiving c_0 and c_1, R uses sk_{R} to compute $M_b \leftarrow \mathsf{SDec}(\mathsf{sk}_{\mathsf{R}}, c_b)$. It outputs M_b and halts.

A scheme of the protocol is presented below. By $x \stackrel{?}{=} y$, we mean that the user checks if x and y have the same value. If they do not have the same value, then the user aborts the protocol. In the scheme, the index i should take the values 0 and 1.

RLWE KE-based OT protocol: π_{OT}

Sender S		**Receiver** R
Input: M_0, M_1		**Input:** $b \in \{0,1\}$
		$s_{\mathsf{R}}, e_{\mathsf{R}}, e'_{\mathsf{R}} \leftarrow_{\$} \chi$
		$p_{\mathsf{R}}^b \leftarrow m s_{\mathsf{R}} + 2e_{\mathsf{R}} \mod q$
$s_{\mathsf{S}}, e_{\mathsf{S}}, e'_{\mathsf{S}} \leftarrow_{\$} \chi$		$r \leftarrow_{\$} \{0,1\}^{\kappa}$
$p_{\mathsf{S}} \leftarrow m s_{\mathsf{S}} + 2e_{\mathsf{S}} \mod q$	$\xleftarrow{(\mathsf{sid}, p_{\mathsf{R}}^0, r)}$	$h \leftarrow \mathsf{H}_1(\mathsf{sid} \mid r)$
$h' \leftarrow \mathsf{H}_1(\mathsf{sid} \mid r)$		$p_{\mathsf{R}}^0 \leftarrow p_{\mathsf{R}}^1 - h \mod q$ (if b=1)
$p_{\mathsf{R}}^1 \leftarrow p_{\mathsf{R}}^0 + h' \mod q$		
$k_{\mathsf{S}}^i \leftarrow s_{\mathsf{S}} p_{\mathsf{R}}^i + 2e'_{\mathsf{S}} \mod q$		
$\sigma_i \leftarrow \mathsf{Sig}(k_{\mathsf{S}}^i)$		
$\mathsf{sk}_{\mathsf{S}}^i \leftarrow \mathsf{Mod}_2(k_{\mathsf{S}}^i, \sigma_i)$		
$w_0, z_0, w_1, z_1 \leftarrow_{\$} \{0,1\}^{\kappa}$		$k_{\mathsf{R}} \leftarrow p_{\mathsf{S}} s_{\mathsf{R}} + 2e'_{\mathsf{R}} \mod q$
$\bar{\mathsf{sk}}_{\mathsf{S}}^i \leftarrow \mathsf{H}_2(\mathsf{sid} \mid \mathsf{sk}_{\mathsf{S}}^i)$		$\mathsf{sk}_{\mathsf{R}} \leftarrow \mathsf{Mod}_2(k_{\mathsf{R}}, \sigma_b)$
$a_i \leftarrow \mathsf{SEnc}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{S}}^i, w_i; z_i)$		$\bar{\mathsf{sk}}_{\mathsf{R}} \leftarrow \mathsf{H}_2(\mathsf{sid} \mid \mathsf{sk}_{\mathsf{R}})$
$\bar{w}_i \leftarrow \mathsf{H}_3(\mathsf{sid} \mid w_i)$		$x_b \leftarrow \mathsf{SDec}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{R}}, a_b)$
$u_i \leftarrow \bar{w}_i \oplus (w_{1-i} \mid \bar{\mathsf{sk}}_{\mathsf{S}}^{1-i} \mid z_{1-i})$	$\xrightarrow{(\mathsf{sid}, p_{\mathsf{S}}, \sigma_0, \sigma_1, a_0, a_1, u_0, u_1)}$	$\bar{x}_b \leftarrow \mathsf{H}_3(\mathsf{sid} \mid x_b)$
$ch \leftarrow \mathsf{H}_4(\mathsf{sid} \mid w_0 \mid w_1 \mid z_0 \mid z_1)$		$(x_{1-b}, \bar{\mathsf{sk}}_{\mathsf{R}}^{1-b}, y_{1-b}) \leftarrow u_b \oplus \bar{x}_b$
		$a_{1-b} \overset{?}{=} \mathsf{SEnc}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{R}}^{1-b}, x_{1-b}; y_{1-b})$
		$\bar{x}_{1-b} \leftarrow \mathsf{H}_3(\mathsf{sid} \mid x_{1-b})$
		$(x'_b, \bar{\mathsf{sk}}_{\mathsf{R}}^b, y_b) \leftarrow u_{1-b} \oplus \bar{x}_{1-b}$
		$x'_b \overset{?}{=} x_b, \quad \bar{\mathsf{sk}}_{\mathsf{R}}^b \overset{?}{=} \bar{\mathsf{sk}}_{\mathsf{R}}$
		$a_b \overset{?}{=} \mathsf{SEnc}_{\Delta}(\bar{\mathsf{sk}}_{\mathsf{R}}^b, x_b; y_b)$
	$\xleftarrow{(\mathsf{sid}, ch')}$	$ch' \leftarrow \mathsf{H}_4(\mathsf{sid} \mid x_0 \mid x_1 \mid y_0 \mid y_1)$
$ch \overset{?}{=} ch'$		
$c_i \leftarrow \mathsf{SEnc}_{\Delta}(\mathsf{sk}_{\mathsf{S}}^i, M_i)$	$\xrightarrow{(\mathsf{sid}, c_0, c_1)}$	$M_b \leftarrow \mathsf{SDec}_{\Delta}(\mathsf{sk}_{\mathsf{R}}, c_b)$

In the protocol, S and R run the RLWE-based KE where R plays the role of Alice. Observe that R cannot play the role of Bob. More precisely, R cannot send a signal of its key, since, in this case, S would be able to tell if R has a secret s_{R} for p_{R}^0 or for p_{R}^1 [DRAC18], thus, revealing b.

The purpose of the challenge sent by the sender to the receiver in the second round is to force the receiver to ask the random oracle with the key it has, in order to perform the simulation when the receiver is corrupted. The simulation when the sender is corrupted is done by programming the random oracle, in order to extract both keys.

By the soundness of the RLWE key exchange, we have that sk_{R} will be equal to $\mathsf{sk}_{\mathsf{S}}^b$. Thus, the receiver will be able to decrypt the ciphertext c_b and recover the message M_b.

A $\binom{N}{1}$-OT protocol. The scheme can be easily extended to create a $\binom{N}{1}$-OT protocol: R sends p_{R}^{0} and also $r_1, \ldots, r_{N-1}$ in the first message. The challenge computed by S is $\mathsf{H}_4(\mathsf{sid}, w_0, \ldots, w_{N-1}, z_0, \ldots, z_{N-1})$ and it sends $a_0, \ldots, a_{N-1}, u_0, \ldots, u_{N-1}$ where

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

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 $i = 0, \ldots, N-1$.

Sender's and receiver's privacy. Note that the receiver is not able to get both messages. Since the hash value h is an uniformly chosen value, then p_{R}^{1-b} is also a uniformly chosen value. Without knowing the secret s such that $p_{\mathsf{R}}^{1-b} = ms + f \mod q$ where $f \leftarrow_{\$} \chi$ (which may not even exist), the receiver is not be able to compute the value $k_{\mathsf{R}}^{1-b} = p_{\mathsf{S}} s + 2e'_{\mathsf{R}} \mod q$ that would allow it to find a key equal to $\mathsf{sk}_{\mathsf{S}}^{1-b}$. This follows immediately from the HNF-RLWE assumption and from the security of the RLWE-based key exchange. Moreover, the receiver gets to know σ_0 and σ_1, where $(p_{\mathsf{S}}, \sigma_i)$ are two messages of the KE protocol, and while the receiver R has the secret key s_{R} for one of them, the other yields no information by the security of the KE (again, since it does not have the secret key for p_{R}^{1-b}). Later, the receiver receives a_0, a_1, u_0, u_1 from the sender from which it is only able to recover the hash of the secret keys. Since the hash functions are modeled as random oracles, the value $\bar{\mathsf{sk}}_{\mathsf{R}}^{1-b}$ will not be correlated with the key $\mathsf{sk}_{\mathsf{S}}^{1-b}$ which will be used to encrypt M_{1-b}. This can be proven using a simple hybrid argument and replacing the hash of $\mathsf{sk}_{\mathsf{S}}^{1-b}$ by a uniformly chosen value. For these reasons, the receiver will not have any information on the secret key $\mathsf{sk}_{\mathsf{S}}^{1-b}$. Hence, it is infeasible for it to get the message M_{1-b}, given that Δ is secure.

Also, we argue that it is infeasible for the sender to know the input b of R. Observe that the sender only receives p_{R}^{0} (from which it can derive p_{R}^{1}) from the receiver, and a random value r. From the HNF-RLWE assumption, it follows that the sender has no information on which of these values, p_{R}^{0} or p_{R}^{1}, was computed using the receiver's secret s_{R} and which will allow it to compute the shared key. The sender S also receives ch', which R can compute no matter its input b.

4 Security proof

The following theorem states the security of the scheme in the UC-framework.

Theorem 4. *The protocol π_{OT} UC-realizes $\mathcal{F}_{OT}$ in the $\mathcal{F}_{RO}$-hybrid model against static malicious adversaries, given that the symmetric-key encryption scheme Δ is secure.*

Proof. In order to prove the theorem stated above, we have to show that both the real-world execution of the protocol and the ideal-world execution of $\mathcal{F}_{\mathrm{OT}}$ are indistinguishable for any environment $\mathcal{E}$.

As usual, the communication with the environment $\mathcal{E}$ can be simulated by Sim in the following way: whenever Sim receives a message from $\mathcal{E}$, Sim forwards it to $\mathcal{A}$; and whenever $\mathcal{A}$ outputs some message, Sim outputs the same message that will be read by $\mathcal{E}$.

In the (trivial) cases where both the sender and the receiver are honest and controlled by the simulator Sim (the simulator just runs the protocol honestly) or when both of them are corrupted by an adversary $\mathcal{A}$ (the simulator just runs internally the adversary which will generate messages from both the corrupted sender and the corrupted receiver) the executions are indistinguishable. We still have to consider the cases where only the receiver is corrupted and when only the sender is corrupted by an adversary. This is done by proving the existence of an ideal-world simulator Sim that mimics the behavior of any real-world adversary $\mathcal{A}$ corrupting only one of the parties.

Security when only the receiver R is corrupted by $\mathcal{A} = \mathcal{A}(\mathsf{R})$. We start by proving security against a corrupted receiver. Given any real-world adversary $\mathcal{A}$ corrupting R (denoted by $\mathcal{A}(\mathsf{R})$), we have to construct a simulator Sim that is able to extract the inputs of $\mathcal{A}(\mathsf{R})$. In this case, Sim must be able to extract the bit b representing the message that R wants to receive. After knowing b, Sim will be able to send it to the ideal functionality. Sim will then receive a message M_b and proceeds the interaction with the real-world adversary sending the message M_b and some other randomly chosen message, in order to complete the protocol. In the end, both executions will be indistinguishable from the point-of-view of the environment.

Given a real-world adversary $\mathcal{A}(\mathsf{R})$, we describe how the simulator Sim works. The adversary can make queries to the random oracles H_1, H_2, H_3 and H_4, here simulated by Sim.

Simulation of H_1*,* H_2*,* H_3 *and* H_4*:* In this case, when $\mathcal{A}(\mathsf{R})$ queries any of these random oracles with $(\mathsf{sid}|t)$, Sim answers as the random oracles would do, returning a value h chosen uniformly at random. It keeps the pair (t, h) of query and respective answer in a list.

At some point, Sim receives $(\mathsf{sid}, p_{\mathsf{R}}^0, r)$ from $\mathcal{A}(\mathsf{R})$. Sim computes k_{S}^i, σ_i and the keys $\mathsf{sk}_{\mathsf{S}}^i \leftarrow \mathsf{Mod}_2(k_{\mathsf{S}}^i, \sigma_i)$, for $i = 0, 1$. It proceeds as the honest sender would do and sends $(\mathsf{sid}, p_{\mathsf{S}}, \sigma_0, \sigma_1, a_0, a_1, u_0, u_1)$ to the adversary.

After sending these values to the adversary, Sim sets $b = \perp$. When $\mathsf{sk}_{\mathsf{S}}^{\bar{b}}$ is asked to the random oracle H_2 by $\mathcal{A}(\mathsf{R})$, it sets $b = \bar{b}$. If w_{1-b} is asked to the random oracle H_3 before w_b, then Sim aborts the execution. If k_{S}^{1-b} is asked to H_2, then it also aborts the execution. At some point, Sim receives ch' from $\mathcal{A}(\mathsf{R})$. It checks if $ch = ch'$: if the test fails, it aborts as the honest S would do. Otherwise, it sends $(\mathsf{sid}|b)$ to the ideal functionality $\mathcal{F}_{\mathrm{OT}}$, which returns back M_b.

Sim computes $c_b \leftarrow \mathsf{SEnc}_\Delta(\mathsf{sk}_\mathsf{S}^b, M_b)$ and $c_{1-b} \leftarrow \mathsf{SEnc}_\Delta(\mathsf{sk}_\mathsf{S}^{1-b}, 0^\lambda)$ where λ is the size of M_b. Finally, it sends c_0 and c_1 to $\mathcal{A}(\mathsf{R})$, waits for it to halt and halts.

Both the real-world and the ideal-world executions are indistinguishable, except when the simulator Sim aborts when it is not supposed to do it. From the specification of Sim, we conclude that it aborts if the key k_S^{1-b} is asked to the random oracle by the adversary, which by the security of the RLWE key exchange scheme used has negligible probability of happening; or if w_{1-b} is asked to the random oracle H_3 before w_b, which happens also with negligible probability on the security parameter κ; or if none of the keys k_S^0 and k_S^1 are queried to the random oracle, but in this case, the receiver will fail to compute the right ch', except with negligible probability. We conclude that

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

for any $\mathcal{A}$ corrupting only R, that is, both executions are indistinguishable.

Security when only the receiver S is corrupted by $\mathcal{A} = \mathcal{A}(\mathsf{S})$. When the real-world adversary is corrupting S, we have to construct a simulator (controlling the dummy receiver) that is able to extract the adversary's input, that is, the simulator must be able to extract the input of the corrupted sender, which are the messages M_0 and M_1.

The simulator proceeds as follows. It starts by choosing $s_\mathsf{R}^0, e_\mathsf{R}^0 \leftarrow_{\$} \chi$ and computes $p_\mathsf{R}^0 \leftarrow m s_\mathsf{R}^0 + 2e_\mathsf{R}^0 \mod q$ and, as the honest R would do, it samples $r \leftarrow_{\$} \{0,1\}^\kappa$ and sends $(\mathsf{sid}, p_\mathsf{R}^0, r)$ to $\mathcal{A}(\mathsf{S})$. Moreover, the simulator chooses $s_\mathsf{R}^1, e_\mathsf{R}^1 \leftarrow_{\$} \chi$ and computes another key $p_\mathsf{R}^1 \leftarrow m s_\mathsf{R}^1 + 2e_\mathsf{R}^1 \mod q$.

The adversary can make queries to the random oracles H_1, H_2, H_3 and H_4. We specify how the simulator simulates each of these random oracles.

Simulation of H_2*,* H_3 *and* H_4*:* Whenever $\mathcal{A}(\mathsf{S})$ queries H_2, H_3 or H_4 with a query $(\mathsf{sid}|t)$, the simulator returns a value h as the ideal functionality would do and keeps (t, h) on a list of queries and respective answers.

Simulation of H_1*:* Whenever the adversary queries H_1 with $(\mathsf{sid}|r)$, the simulator answers with $h = p_\mathsf{R}^1 - p_\mathsf{R}^0$, in all other cases it returns h as the ideal functionality would do, and again keeps (t, h) on a list of queries and respective answers. Note that the value h for the query (sid, r) is computationally indistinguishable from a uniformly chosen value since $p_\mathsf{R}^1 - p_\mathsf{R}^0$ is indistinguishable from random given that the HNF-RLWE assumption holds. Consequently, the probability that the environment distinguishes both the real and the simulated execution of the random oracle H_1 is negligible.

At some point, Sim receives $(\mathsf{sid}, p_\mathsf{S}, \sigma_0, \sigma_1, a_0, a_1, u_0, u_1)$ from $\mathcal{A}(\mathsf{S})$. Note that, from the information $(p_\mathsf{S}, \sigma_0, \sigma_1)$, the simulator can compute both

$$k_\mathsf{R}^0 \leftarrow p_\mathsf{S} s_\mathsf{R}^0 + 2e'_\mathsf{R} \mod q \quad \text{and} \quad k_\mathsf{R}^1 \leftarrow p_\mathsf{S} s_\mathsf{R}^1 + 2e'_\mathsf{R} \mod q$$

and the keys

$$\mathsf{sk}_\mathsf{R} \leftarrow \mathsf{Mod}_2(k_\mathsf{R}^0, \sigma_0) \quad \text{and} \quad \mathsf{sk}_\mathsf{R} \leftarrow \mathsf{Mod}_2(k_\mathsf{R}^1, \sigma_1)$$

that should be equal to the keys computed by $\mathcal{A}(\mathsf{S})$. Sim proceeds as the honest R would do and sends ch' to the $\mathcal{A}(\mathsf{S})$.

$\mathcal{A}(\mathsf{S})$ sends c_0 and c_1 to the dummy R. Since Sim has both keys, it can compute

$$M_0 \leftarrow \mathsf{SDec}_\Delta(\mathsf{sk}_\mathsf{R}^0, c_0) \quad \text{and} \quad M_1 \leftarrow \mathsf{SDec}_\Delta(\mathsf{sk}_\mathsf{R}^1, c_1).$$

It sends both $(\mathsf{sid}|M_0|M_1)$ to the ideal functionality $\mathcal{F}_{\mathrm{OT}}$.

Finally, upon receiving a message $(\mathsf{sid}|\mathit{receipt})$ from the ideal functionality, Sim waits for the adversary to halt and halts.

We have to show that both the ideal-world and the real-world executions are indistinguishable from the point-of-view of any environment. The ideal-world simulation just differs from the real-world execution in the outputs of the random oracle H_1. As mentioned above, it is infeasible for an adversary (or environment) to distinguish the value h outputted by H_1 when queried on r from a truly execution of the random oracle, by the HNF-RLWE assumption. Hence, we conclude that

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

for any adversary $\mathcal{A}$ corrupting S. □

5 Efficiency

In this section, we analyze the communication and computational complexity of the OT scheme presented in this paper. We also compare it with other post-quantum UC-secure protocols. Finally, we propose some optimizations that should be considered when implementing the protocol in real-life.

Let λ be the length of the messages that S wants to obviously send to R, Δ be the usual one-time pad and κ be the security parameter. Note that the most expensive part of our protocol is the key exchange since all the other operations (sum modulo q, sum modulo 2 and concatenation) are linear on the number of bits.

Recall that the communication complexity of the RLWE-based KE is $2\mu \log q + \mu$ and that it requires four multiplications in the ring R_q where μ is the length of the key exchanged [DXL12]. Since we are using one-time pad $\mu = \lambda$.

Communication Complexity. The first two rounds of the OT protocol are almost the same as the RLWE-based KE. Hence, the communication complexity of the OT protocol is $2\lambda \log q + \lambda$ (corresponding to the execution of the RLWE-based KE) plus κ (corresponding to the length of r) plus 8κ bits of information, where 2κ bits corresponds to (a_0, a_1) and $2(3\kappa)$ bits to (u_0, u_1).[1] This results in $2\lambda \log q + \lambda + 9\kappa$ bits of information. The third message exchanged is just a κ-bit message, which is the answer to the challenge. Finally, the fourth message, composed by two ciphertexts (c_0, c_1), requires 2λ bits of information. So in total, our scheme requires $2\lambda \log q + 3\lambda + 10\kappa$ bits of information to be exchanged.

Computational complexity. The scheme just requires five multiplications in the ring R_q and six samplings from the error distribution χ, the same as in the RLWE-based KE. Therefore, our scheme is very efficient in terms of computational complexity, requiring only these operations to obliviously transfer a λ-bit message.

Table 1: Number of rounds, communication complexity and computation complexity for obliviously transmitting a λ-bit message using π_{OT}, with security parameter κ. Computational complexity is expressed in "number of multiplications in the ring R_q"+"number of samplings from the distribution χ".

N. of rounds	Communication complexity	Computation complexity
4	$2\lambda \log q + 3\lambda + 10\kappa$	$5 + 6$

Comparison. The scheme of Peikert *et al.* [PVW08] (and the one in [BPRS17]) can be implemented using a dual-mode version of the LWE public-key encryption scheme [Reg05] and it is not known if it can be instantiated with a version of the RLWE encryption scheme. In fact, it is stated as an open problem in [LKHB17] to find a dual-mode RLWE encryption scheme. The dual-mode LWE public-key encryption scheme used in the OT requires a public key of size $\mathcal{O}(n^2)$ and the encryption takes $\mathcal{O}(n \log n)$ operations, for a single bit message [PVW08], where n depends on the security parameter. The framework requires a key generation, two encryptions and a decryption. To transfer a λ-bit message, this has to be repeated λ times. Let us assume that $\lambda \approx n$. [2]

In terms of the communication complexity, our scheme has two more rounds than [PVW08]. However, the communication complexity of our scheme is linear in λ and, thus, it requires less information to be exchanged asymptotically. Note that, in the scheme of [PVW08], a public key needs to sent from the receiver to the sender requiring $\mathcal{O}(\lambda^2)$ bits of information to be exchanged, just for a single bit to be transfered. For a λ-bit message to be transfered, the scheme of [PVW08] needs $\mathcal{O}(\lambda^3)$ bits of information to be exchanged, just in the first round.

[1] Here, we just need a κ-bit secret key to use the one-time pad on a κ-bit message.

[2] Usually, the objective of OT is to exchange a key in order to perform multi-party computation [Yao86]. By [Alb17], we conclude that n should be around 1024, and, thus, let us suppose that the message being obliviously sent is of the same size. The same assumption was done while analyzing the efficiency of our scheme.

Regarding the computational cost, their scheme requires the generation of one pair of public and secret keys of the LWE public-key encryption scheme, the encryption of two messages and the decryption of one ciphertext. It is well known that primitives based on the RLWE assumption are faster and require less memory than the ones based on the LWE assumption. This fact is also reflected here. [3] The computational cost of the scheme of [PVW08] is $\mathcal{O}(\lambda^3)$, corresponding to the key generation, plus two encryptions, which cost $\mathcal{O}(\lambda^2 \log^2 \lambda)$ operations. Whereas, our scheme just requires five multiplications in the ring R_q, which, using Fast Fourier Transform, takes $\mathcal{O}(\lambda \log q)$ operations for polynomials of degree λ. Therefore, although our scheme has two more rounds than the one in [PVW08], the operations of our scheme are faster and the size of the messages exchanged are shorter, resulting in an overall asymptotically more efficient procedure.

We also would like to remark that the scheme in [PVW08] is proven to be secure in the Common Reference String (CRS) model. Of course, this raises the question on how to create the common reference string in practice. Either we delegate this operation to a (trusted) third party or we use some multi-party computation procedure to create the common reference string, which would be too inefficient. Our scheme relies on random oracles that can be instantiated using cryptographic hash functions in practice and, thus, it should be very easy and efficient to implement.

It seems that it is not possible to instantiate the scheme of Barreto *et al.* [BDD$^+$17] using LWE or the RLWE public-key encryption scheme since a group structure is required in the set of public keys [BDD$^+$17, Property 2.2]. However, the set of LWE and RLWE public keys does not have a group structure for any operation. For example, if we had two different samples of the RLWE distribution, the sum will not be a RLWE sample. Thus, the conditions presented in [BDD$^+$17] discard the use of LWE or RLWE public-key encryption schemes in their framework.

Possible optimizations. Some steps and procedures of the protocol may be optimized in practice. For instance, the polynomial m can be the same in multiple iterations of the protocol.[4] The value p_{S} sent by the sender in the first round can be used in several iterations of the protocol, taking into account that security may be compromised, when used too many times [Flu16, DAS$^+$17, DFR18].

Another possible optimization is to use a more practical symmetric-key encryption scheme than the one-time pad. Indeed, AES can be computed using specialized hardware which makes it extremely fast comparing to public-key encryption schemes. It also allows smaller keys, which removes the practical issues associated with the use of one-time pad.

[3] An analysis of the number of operations of LWE encryption scheme and RLWE Ding's KE is presented in [DXL12]

[4] In fact, it is suggested in [DXL12] that m is chosen by some trusted institution (e.g. NIST) and used as a standard.

6 Conclusion

To the best of our knowledge, we present the first ever universally composable OT protocol whose security is based on the RLWE assumption. Its simplicity and the fact that is based on the RLWE problem allows for very efficient computational and communication complexity. For these reasons, we believe that it is a solid candidate to be employed in post-quantum secure applications. We also proved UC-security for the proposed protocol, which means that it can be composed arbitrarily with the same or other protocols.

In real-life, the protocol should be quite efficient. We suggest to use the NIST standard SHA to model random oracles. Regarding encryption, one may use AES (which is also the current NIST standard) instead of the one-time pad, which leads to a much smaller key size. These standards are chosen considering the fact that they should be easily and efficiently implemented in hardware, leading to extremely fast executions. Moreover, attacks to break these protocols using a quantum computer are not known. The implementation in hardware of our protocol and comparison with other OT protocols are left as future work.

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 grant 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.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret lwe and parameter choices in helib and seal. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 103–129, Cham, 2017. Springer International Publishing.

[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`.

[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.

[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.

[DAL+17] Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 183–204, Cham, 2017. Springer International Publishing.

[DAS+17] J. Ding, S. Alsayigh, R. V. Saraswathy, S. Fluhrer, and X. Lin. Leakage of signal function with reused keys in RLWE key exchange. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[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.

[DFR18] Jintai Ding, Scott Fluhrer, and Saraswathy R.V. Complete attack on RLWE key exchange with reused keys, without signal leakage.

In Willy Susilo and Guomin Yang, editors, *Information Security and Privacy*, pages 467–486, Cham, 2018. 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 *IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 595–612, 2018.

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

[DRAC18] Jintai Ding, Saraswathy RV, Saed Alsayigh, and Crystal Clough. How to validate the secret of a ring learning with errors (rlwe) key. Cryptology ePrint Archive, Report 2018/081, 2018. https://eprint.iacr.org/2018/081.

[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.

[Flu16] Scott Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016. https://eprint.iacr.org/2016/085.

[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.

[LH18] Momeng Liu and Yupu Hu. Universally composable oblivious transfer from ideal lattice. *Frontiers of Computer Science*, Oct 2018.

[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.

[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.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[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.

[ZZD+15] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 719–751, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.