Universally Composable Oblivious Transfer based on a variant of LPN

Bernardo David[1]*, Rafael Dowsley[2], and Anderson C. A. Nascimento[3]

[1] Department of Computer Science
Aarhus University
bernardo@cs.au.dk

[2] Institute of Theoretical Informatics
Karlsruhe Institute of Technology
rafael.dowsley@kit.edu

[3] Department of Electrical Engineering
University of Brasilia
andclay@ene.unb.br

Abstract. Oblivious transfer (OT) is a fundamental two-party cryptographic primitive that implies secure multiparty computation. In this paper, we introduce the first OT based on the Learning Parity with Noise (LPN) problem. More specifically, we use the LPN variant that was introduced by Alekhnovich (FOCS 2003). We prove that our protocol is secure against active static adversaries in the Universal Composability framework in the common reference string model. Our constructions are based solely on a LPN style assumption and thus represents a clear next step from current code-based OT protocols, which require an additional assumption related to the indistinguishability of public keys from random matrices. Our constructions are inspired by the techniques used to obtain OT based on the McEliece cryptosystem.

1 Introduction

Oblivious transfer (OT) [46, 44, 21] was introduced in the early days of public-key cryptography and has thereafter played an essential role in modern cryptography. They imply, among other things, the possibility of performing two-party secure computation [24, 34] and multi-party computation [13]. Initially many variants of OT were considered, but they are equivalent [12] and therefore in this work we will focus on the most common one: one-out-of-two bit oblivious transfer. In this variant there is a sender who inputs two bits x_0 and x_1, and a receiver who chooses which bit x_c he wants to learn. On one hand, the receiver should learn x_c, but should have no information about $x_{\overline{c}}$. On the other hand, the sender should not learn the choice bit c.

Given the importance of OT protocols, constructions were extensively studied and nowadays solutions are known based on both generic computational assumptions such as enhanced trapdoor permutations [21], and also based on specific computational assumptions such as: the hardness of factoring [44, 27], the Decisional Diffie-Hellman (DDH) assumption [4, 39, 1, 47], the Quadratic Residuosity (QR) assumption [27], the N'th residuosity assumption [27], the hardness of the Subgroup Decision Problem [36], and the McEliece assumptions [19]. Since Shor's algorithm [45] would make factoring and computing discrete logarithms easy in the case that quantum computers become practical, an important question is determining which post-quantum assumptions are sufficient to implement OT protocols. LPN-based/code-based cryptography is one of the main alternatives for a post-quantum world and thus our result improves the understanding in this area.

As with most cryptographic primitives, the first OT protocols considered simple security models (in this case the stand alone model in which there is only one execution of the protocol isolated from the rest of the world). Afterwards, stronger models were considered, such as security in the Universal

* Supported by European Research Council Starting Grant 279447.

Composability (UC) framework by Canetti [5], which allows arbitrary composition of the protocols. This latter notion is the most desirable security goal for oblivious transfer protocols, since it allows these protocols to be used as building blocks of more complex primitives and protocols.

In this work we will present the first OT protocol based on a variant of the Learning Parity with Noise (LPN) problem that was introduced by Alekhnovich [2, 3]. The protocol achieves UC security against active static adversaries following ideas similar to the ones that Dowsley et al. [19, 20, 15] used to build OT protocols based on the McEliece assumptions [37]. It is well-known that UC-secure oblivious transfer is impossible in the plain model [6, 7], so our solution is in the common reference string (CRS) model.

1.1 Related Works

Cryptography based on Codes and LPN: McEliece [37] proposed a cryptosystem based on the hardness of the syndrome decoding problem. Later on, Niederreiter [40] proposed a cryptosystem that is the dual of McEliece's cryptosystem. These cryptosystems can be modified to achieve stronger notions of security such as IND-CPA [41, 42] and IND-CCA2 [18, 22, 16]. Based on these cryptosystems it is possible to implement both stand alone secure [19, 20] and UC-secure [15] OT protocols. The main drawback of these code-based schemes is that, besides assuming the hardness of the decoding problem, they also assume that the adversary is not able to recover the hidden structure of the keys, which is formalized by assuming that the public-keys are indistinguishable from random matrices. But this later problem is far less studied than the decoding one.

Building public-key encryption schemes from the original LPN problem is a difficult task and so far the only schemes are based on a variant of the LPN problem introduced by Alekhnovich in [2, 3], which yields semantically secure encryption [2, 3, 30] and IND-CCA2 secure encryption by Döttling et al. [17]. Moreover, other cryptographic primitives were built based solely on the Alekhnovich variant of the LPN problem, such as: pseudo random generators (PRG) [30], message authentication codes (MAC) [30], pseudo random functions (PRFs) [30], signature schemes with constant overhead [30], zero-knowledge [31], and commitments [31].

Furthermore, Ishai *et al.* present a protocol for secure two-party and multiparty computation with constant computational overhead in the semi-honest model and slightly superlinear computational overhead in the malicious model based on Alekhnovich's LPN [30]. However, their secure computation constructions assume the existence of bit oblivious transfer, which wasn't built from Alekhnovich's LPN until now (not even with stand-alone security).

Universally Composable OT: Peikert et al. developed a general framework for obtaining efficient, round optimal UC-secure OT in the CRS model [43] that provides instantiations based on the DDH, QR and Learning With Errors (LWE) [43]. Constructions of OT protocols that achieve UC security against different kinds of adversaries under various setup assumptions are also known to be possible under the Decisional Linear (DLIN) assumption [14, 32], the DDH and the strong RSA assumptions [23] and the Decisional Composite Residuosity (DCR) assumption [32, 11].

Another approach to obtain UC-secure oblivious transfer protocols is to take a stand alone secure OT protocol and use compilers [29, 26, 10] to achieve an UC-secure protocol. However these compilers require access to UC-secure string commitment schemes that were not yet built from the LPN assumption.

1.2 Our Contributions

In this work we address the open problem of constructing oblivious transfer based on the assumption that LPN is hard. We focus on the LPN variant introduced by Alekhnovich in [2, 3]. Our main result is the first Oblivious Transfer protocol based on LPN. Our protocol is Universally Composable and offers security against active static adversaries, *i.e.* adversaries that may deviate in any arbitrary way from the protocol but are forced to corrupt their desired parties before protocol execution starts. It is well-known that UC realizing any interesting multiparty functionality (among them OT) is impossible in the plain model (*i.e.* without a setup assumption) [6, 7]. Hence, we build our protocol in the Common Reference String (CRS) model, where the parties are assumed to have access to a fixed string generated before protocol execution starts.

The protocol is based on the cut-and-choose approach of [15], although with a different proof strategy. This approach basically requires a stand-alone passively secure OT protocol and an extractable commitment scheme as building blocks. We show that a stand alone OT protocol (with passive or active security) can be obtained in a similar way as in [19, 20]. We also observe that we can obtain an extractable commitment scheme from an IND-CPA secure public key encryption scheme based on Alekhnovich's LPN assumption introduced in [17].

Besides proving that it is possible to construct oblivious transfer from variants of the LPN assumption, our results greatly improve on previous code-based OT protocols by relying on a weaker assumption. Moreover, together with the CCA2 secure Alekhnovich cryptosystem [17] and the LPN based proofs of knowledge and commitments [31], our results contribute towards obtaining more complex cryptographic protocols based on coding based assumptions weaker than McEliece.

1.3 Outline

In Section 2 we introduce the notation, assumptions and definitions used throughout the paper. In Section 3, we present a basic stand alone OT protocol that will serve as a building block. In Section 4, we present the active secure universally composable OT protocol based on cut-and-choose techniques. Finally, in Section 5, we conclude with directions for future research.

2 Preliminaries

In this section we introduce our notation and also recall the relevant definitions.

2.1 Notation

If x is a string, then $|x|$ denotes its length, while $|\mathcal{X}|$ represents the cardinality of a set $\mathcal{X}$. If $n \in \mathbb{N}$ then 1^n denotes the string of n ones. $s \leftarrow S$ denotes the operation of choosing an element s of a set S uniformly at random. $w \leftarrow \mathcal{A}^{\mathcal{O}}(x, y, \ldots)$ represents the act of running the algorithm $\mathcal{A}$ with inputs $x, y, \ldots$, oracle access to $\mathcal{O}$ and producing output w. $\mathcal{A}^{\mathcal{O}}(x; r)$ denotes the execution with coins r. We denote by $\Pr(E)$ the probability that the event E occurs. If a and b are two strings of bits or two matrices, we denote by $a|b$ their concatenation. The transpose of a matrix M is M^T. If a and b are two strings of bits, we denote by $\langle a, b \rangle$ their dot product modulo 2 and by $a \oplus b$ their bitwise XOR. $\mathcal{U}_n$ is an oracle that returns an uniformly random element of $\{0,1\}^n$. If b is a bit, then $\overline{b}$ denotes its inverse (*i.e.* $1 - b$). Let $\mathbb{F}_2$ denote the finite field with 2 elements. For a parameter ρ, χ_ρ denotes the Bernoulli distribution that outputs 1 with probability ρ.

2.2 Encryption Scheme

In this section we describe the LPN-based public-key encryption scheme that was introduced by Döttling et al. [17] and that will be used in this paper. Note that we use the simplest version of their cryptosystem, the one which only achieves IND-CPA security (which is already enough for our purposes) and does not allow witness recovery.

Let n be the security parameter, $n_1, \ell_1, \ell_2 \in O(n^{2/(1-2\epsilon)})$, and $\rho \in O(n^{-(1+2\epsilon)/(1-2\epsilon)})$. Let $G \in \mathbb{F}_2^{\ell_2 \times n_1}$ be the generator-matrix of a binary linear error-correcting code $\mathcal{C}$ and $\mathsf{Decode}_{\mathcal{C}}$ an efficient decoding procedure for $\mathcal{C}$ that corrects up to $\alpha\ell_2$ errors for a constant α.

Key Generation: Sample a uniformly random matrix $A \in \mathbb{F}_2^{\ell_1 \times n_1}$, a matrix T from $\chi_\rho^{\ell_2 \times \ell_1}$ and a matrix X from $\chi_\rho^{\ell_2 \times n_1}$. Set $B = TA + X$. Set $\mathsf{pk} = (A, B, G)$ and $\mathsf{sk} = T$. Output $(\mathsf{pk}, \mathsf{sk})$.

Encryption $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$: Given a message $\mathsf{m} \in \mathbb{F}_2^{n_1}$ and the public key $\mathsf{pk} = (A, B, G)$ as input, sample s from $\chi_\rho^{n_1}$, e_1 from $\chi_\rho^{\ell_1}$ and e_2 from $\chi_\rho^{\ell_2}$. Then set $\mathsf{ct}_1 = As + e_1$ and $\mathsf{ct}_2 = Bs + e_2 + G\mathsf{m}$. Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

Decryption $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Given a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and a secret key $\mathsf{sk} = T$ as input, compute $y = \mathsf{ct}_2 - T\mathsf{ct}_1$ and $\mathsf{m} = \mathsf{Decode}_{\mathcal{C}}(y)$. Output m.

The IND-CPA security of this scheme was proved under the following assumption which is equivalent to Alekhnovich's hardness assumption [17].

Assumption 1 *Let $n_1 \in \mathbb{N}$ be the problem parameter, $m = O(n_1)$, $\epsilon > 0$ and $\rho = \rho(n_1) = O(n_1^{-1/2-\epsilon})$. Choose uniformly at random $A \in \mathbb{F}_2^{m \times n_1}$ and $x \in \mathbb{F}_2^{n_1}$. Sample e according to χ_ρ^m. The problem is, given A and $y \in \mathbb{F}_2^m$, to decide whether y is distributed according to $Ax + e$ or uniformly at random.*

The current best algorithms to attack this problem require time of the order $2^{n^{1/2-\epsilon}}$ and for this reason by setting $n_1 = O(n^{2/(1-2\epsilon)})$ where n is the security parameter of the encryption scheme the hardness is normalized to $2^{\Theta(n)}$.

2.3 Extractable Commitment Schemes

A string commitment scheme is said to be *extractable* if there exists a polynomial-time simulator that is able to obtain the committed value m before the *Open* phase. In the CRS model, we will build an extractable commitment scheme based on the encryption scheme from the previous section in the following way. The CRS contains a public key pk of the cryptosystem and the scheme works as follows:

– $\mathsf{Com}_{\mathsf{crs}}(\mathsf{m})$ The sender encrypts m under the public key pk with randomness (s, e_1, e_2) and sends the corresponding ciphertext ct to the receiver as a commitment.
– $\mathsf{Open}_{\mathsf{crs}}(\mathsf{m})$ The sender sends the message m and the randomness (s, e_1, e_2) used in the commitment phase. The receiver checks if the encryption of m with the randomness (s, e_1, e_2) results in the ciphertext ct that he received before. Additionally, for a fixed constant $\gamma > 1$ such that $\gamma\rho < \alpha/3$, he checks if the Hamming weights of s, e_1 and e_2 are respectively smaller than $\gamma\rho n_1$, $\gamma\rho\ell_1$ and $\gamma\rho\ell_2$. If all tests are passed, the receiver accepts the opening as correct.

Note that in the case that both parties are honest, the Hamming weight tests will be passed with overwhelming probability, as it was shown in the proof of the cryptosystem [17] that larger Hamming weights only occur with negligible probability, so the correctness of the commitment scheme follows. The hiding property follows trivially from the IND-CPA security of the encryption scheme. For the binding property, first notice that the Hamming weight tests performed during the opening phase ensure that the error term $Xs + e_2 - Te_1$ that would appear in a decryption operation of $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; s, e_1, e_2)$ would be within the decoding limit of $\mathcal{C}$ and so the decryption would have been successfully performed and m recovered (see the proof of correctness of [17] for details). I.e., for any opening information $(\mathsf{m}, s, e_1, e_2)$ that passes the tests, we have $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; s, e_1, e_2)) = \mathsf{m}$ and $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; s, e_1, e_2) = \mathsf{ct}$. Thus, due to the uniqueness of the decryption, there is only one m that can pass all the tests performed in the opening phase.

In order to extract the committed values, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk})$ for the cryptosystem and sets the CRS to pk. With the knowledge of the secret key sk, he can extract from any ct the only value m that can be successfully opened in a later stage.

2.4 Universal Composability

The Universal Composability framework was introduced by Canetti in [5] to analyze the security of cryptographic protocols and primitives under arbitrary composition. In this framework, protocol security is analyzed by comparing an ideal world execution and a real world execution. The comparison is performed by an *environment* $\mathcal{Z}$, which is represented by a *PPT* machine and has direct access to all inputs and outputs of the individual parties and to the adversary $\mathcal{A}$. In the ideal world execution, dummy parties (possibly controlled by a *PPT simulator* $\mathcal{S}$) interact directly with the ideal functionality $\mathcal{F}$, which works as trusted third party that computes the desired function or primitive. In the real world execution, several *PPT* parties (possibly corrupted by a real world adversary $\mathcal{A}$) interact with each other by means of a protocol π that realizes the ideal functionality. The real world execution is represented by the ensemble $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$, while the ideal execution is represented by the $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ (that represents all the things that happen outside of the protocol execution) is not able to efficiently distinguish between $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$ and $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, thus implying that the real world protocol is as secure as the ideal functionality. Security in the UC framework is formally defined as:[4]

Theorem 1. *A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ if, for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that, for every PPT environment $\mathcal{Z}$, the following holds:*

$$EXEC_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$$

Adversarial Model In this work we consider security against static adversaries, *i.e.* the adversary corrupts parties before the protocol execution and corrupted parties remain so during the whole execution. Moreover, we consider active adversaries, which may arbitrarily deviate from the protocol in order to perform an attack.

Setup Assumptions The security of our protocol is proved in the Common Reference String (CRS) model (referred to as the $\mathcal{F}_{CRS} - hybrid$ model in [5]), where protocol parties are assumed to have access to a fixed string generated according to a specific distribution before protocol execution starts, in a so called setup phase. The CRS ideal functionality $\mathcal{F}_{CRS}$ is formally presented below.

Common Reference String Ideal Functionality The formal definition of the CRS ideal functionality $\mathcal{F}^{\mathcal{D}}_{CRS}$ is taken from [9].

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

$\mathcal{F}^{D}_{CRS}$ runs with parties $(P_1, ..., P_n)$ and is parametrized by an algorithm $\mathcal{D}$.

• When receiving a message (sid, P_i, P_j) from P_i, let $\mathsf{crs} \leftarrow \mathcal{D}(1^n)$, send $(\mathtt{sid}, \mathsf{crs})$ to P_i and send (crs, P_i, P_j) to the adversary. Next, when receiving $(\mathtt{sid}, P_i, P_j)$ from P_j (and only P_j), send $(\mathtt{sid}, \mathsf{crs})$ to P_j and to the adversary, and halt.

Oblivious Transfer Ideal Functionality The basic 1-out-of-2 oblivious transfer functionality $\mathcal{F}_{OT}$ as defined in [8] is presented bellow.

[4] For the sake of brevity, we refer the reader to Canetti's work [5] for further details and definitions regarding the UC framework.

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

$\mathcal{F}_{OT}$ interacts with a sender **S** and a receiver **R**.

• Upon receiving a message $(\texttt{sid}, \texttt{sender}, x_0, x_1)$ from **S**, where each $x_i \in \{0,1\}^\ell$, store (x_0, x_1) (the length of the strings is fixed and known to all parties).

• Upon receiving a message $(\texttt{sid}, \texttt{receiver}, c)$ from **R**, check if a $(\texttt{sid}, \texttt{sender}, \cdots)$ message was previously sent. If yes, send $(\texttt{sid}, x_c)$ to **R**, $\texttt{sid}$ to the adversary $\mathcal{S}$ and halt. If not, send nothing to **R** (but continue running).

Similarly to the framework of [43], our protocols reuse the same CRS for multiple oblivious transfer invocations. In order to achieve this, we employ the same techniques of [43]. Namely, we "wrap" each single execution of $\mathcal{F}_{OT}$ with a multi session extension $\hat{\mathcal{F}}_{OT}$ of the OT functionality, which handles the multiple independent execution of the OT protocol and coordinates the interaction with parties. The security of the resulting protocol is implied by the UC theorem with joint state (JUC) [9], which states that any protocol π operating in the $\mathcal{F}_{OT} - \textit{hybrid}$ model can be securely emulated in the real world by appropriately composing π with a single execution of a protocol ρ that implements $\hat{\mathcal{F}}_{OT}$.

3 Basic Stand Alone OT Protocol

We first present a basic version of our protocol that is secure only against passive adversaries, then in section 4 we will improve the protocol in order to make it UC-secure against active adversaries. Our construction builds on a IND-CPA secure cryptosystem following the paradigm of [4], previously employed in [19] to obtain OT based on the McEliece assumptions. **R** sends two keys to **S**, one is a proper public-key for which he knows the corresponding secret-key and the other is a "random key" for which he does not know the corresponding trapdoor (in our case this is just a random matrix, for which the decoding problem is assumed to be hard). The keys should be such that **S** cannot tell them apart, which is the case in our protocol. When in possession of the keys, **S** encrypts x_0 and x_1 using the respective keys and sends the ciphertexts to **R**. **R** can recover the bit that was encrypted using the key for which he knows the secret-key, but not the other bit since the decoding problem is assumed to be difficult.

Let the inputs of **S** be the bits x_0 and x_1 and c be **R**'s choice bit. The OT protocol based on the LPN-based IND-CPA secure cryptosystem described in Section 2.2 works as follows.

Protocol 1

1. **R** runs the key generation algorithm to obtain a public-key $\mathsf{pk} = (A, B, G)$ and a secret-key sk. He sets $\mathsf{pk}_c = \mathsf{pk}$, samples uniformly random matrices of the same size as the public key matrices $A' \in \mathbb{F}_2^{\ell_1 \times n_1}$, $B' \in \mathbb{F}_2^{\ell_2 \times n_1}$, $G' \in \mathbb{F}_2^{\ell_2 \times n_1}$ and sets $\mathsf{pk}_{\overline{c}} = (A \oplus A', B \oplus B', G \oplus G')$. Finally he sends pk_0 and pk_1 to **S**.
2. **S** encrypts the messages $x_0 \in \mathbb{F}_2^{n_1}$ and $x_1 \in \mathbb{F}_2^{n_1}$ using the keys pk_0 and pk_1, respectively, and sends the ciphertexts ct_0 and ct_1 to **R**.
3. **R** decrypts ct_c using the secret-key sk and outputs x_c.

The next theorem formally states the security of the above protocol.

Theorem 2. *Protocol 1 is complete and secure for both* **S** *and* **R** *against passive attacks under Assumption 1.*

Proof. Passive adversaries always follow the protocol instructions, so the completeness and security of the protocol can be proved as follows.

Completeness: Since $\mathbf{R}$ has the corresponding secret key and the cryptosystem is itself complete, $\mathbf{R}$ can recover the bit x_c.

Security for S: Let $\widetilde{\mathbf{R}}$ be an honest-but-curious receiver. Note that $\mathsf{pk}_{\overline{c}}$ is completely random and for such keys the encryptions of x_0 and x_1 are computationally indistinguishable from random strings and thus $\widetilde{\mathbf{R}}$ cannot learn any information about $x_{\overline{c}}$ [17].

Security for R: It was proved in [31, 17] that the public-key of the cryptosystem is pseudorandom. Hence $\mathbf{S}$ cannot distinguish the pseudorandom key pk_c from the completely random key $\mathsf{pk}_{\overline{c}}$. Since these are the only information that she receives, she cannot learn anything about $\mathbf{R}$'s choice.

3.1 Obtaining an Stand Alone Active Secure OT Protocol

Notice that, if $\mathbf{R}$ is allowed to deviate from Protocol 1, he could generate another valid public-key pk' and set $\mathsf{pk}_{\overline{c}} = \mathsf{pk}'$. Hence, he would be able to decrypt both bits x_0 and x_1, breaking the security for $\mathbf{S}$. Moreover, the issue of composability and concurrent execution is not addressed by Protocol 1. Thus, the protocol presented in the previous section is only secure in the stand alone case against semi-honest adversaries.

The next step for obtaining active secure universally composable OT consists in constructing an active secure protocol in the stand alone model. In order to achieve this, we can apply techniques from the active secure stand alone protocol in [19] to convert Protocol 1 into an active secure protocol. As in the protocol from [19] this is done by adding parallel repetition and a cut-and-choose phase to the semi-honest protocol in order to guarantee that $\mathbf{R}$ is following the protocol with high probability. For the sake of brevity, we will provide a brief description of the final active secure stand alone protocol, we refer interested readers to [19] for the detailed description.

The main idea to obtain security against a malicious adversary is to have $\mathbf{S}$ control the random matrices (A', B', G') that are added to the valid key. The protocol starts by requiring $\mathbf{R}$ to commit to its valid key pk_c before seeing (A', B', G'). After the commitment, $\mathbf{S}$ sends (A', B', G') to $\mathbf{R}$, who computes a scrambled key $\mathsf{pk}_{\overline{c}}$. $\mathbf{R}$ sends only $\mathsf{pk}_1 = (\tilde{A}, \tilde{B}, \tilde{G})$ to $\mathbf{S}$, who is able to compute $\mathsf{pk}_0 = (\tilde{A} \oplus A', \tilde{B} \oplus B', \tilde{G} \oplus G')$. Finally, $\mathbf{S}$ can either complete the protocol by using $\mathsf{pk}_0, \mathsf{pk}_1$ to encrypt and send its messages to $\mathbf{R}$ or require $\mathbf{R}$ to open its commitment to pk_c (therefore using this instance as a test instance). If $\mathbf{R}$ opens its commitments, $\mathbf{S}$ is able to check whether either pk_0 or pk_1 is equal to the committed value, which means that $\mathbf{R}$ is following the protocol.

Active security can be obtained by executing several randomized copies of the above protocol in parallel and performing a cut-and-choose phase. In this construction, $\mathbf{R}$ runs each parallel instance of the protocol using a random choice bit d_i, which results in the execution of several random OTs. In the cut-and-choose phase, $\mathbf{S}$ requires $\mathbf{R}$ to open half of its commitments to valid public-keys and verifies if $\mathbf{R}$ was following the protocol for those instances. If $\mathbf{S}$ does not detect any cheating, $\mathbf{R}$ derandomizes the instances related to the unopened commitments and $\mathbf{S}$ completes the transfer. Notice, that $\mathbf{S}$ does not learn anything about $\mathbf{R}$'s choice bit, since the opened commitments are related to randomized OTs.

4 Universally Composable Active Secure OT

In this section, we construct an universally composable OT protocol secure against active static adversaries in the Common Reference String model. Using cut-and-choose techniques similar to [15] we depart from a stand alone OT protocol described in Section 3. However, an extractable commitment is employed instead of regular commitment scheme, allowing the simulator to cheat and obtain necessary information to carry out the simulation.

In the universally composable protocol, the receiver $\mathbf{R}$ generates a number of valid public keys K_{i,d_i} for random d_i's and commits to them. In contrast to [15], both players run a coin tossing protocol to generate the random paddings R_i that are used by $\mathbf{R}$ to scramble each valid public key as $K_{i,\overline{d}_i} = K_{i,d_i} + R_i$. $\mathbf{R}$ sends all $K_{i,1}$ keys to the sender $\mathbf{S}$, who retrieves keys $K_{i,0} = K_{i,1} + R_i$. Next, another coin tossing protocol is run between $\mathbf{S}$ and $\mathbf{R}$ to obtain a random string Ω. For each bit equal to 1 in Ω, $\mathbf{R}$ opens the corresponding commitments to valid public keys for verification, as described in Section 3. For each bit equal to 0 in Ω, $\mathbf{R}$ sends to $\mathbf{S}$ information that derandomizes the corresponding public key pairs such that the valid public key corresponds to his choice bit. $\mathbf{S}$ uses the corresponding public key pairs to encrypt an additive share of its messages such that $\mathbf{R}$ can only retrieve a message if it's able to decrypt all ciphertexts.

We use the LPN-based IND-CPA secure public key cryptosystem from [17] (described in Section 2.2) as a building block for encryption and extractable commitments (described in Section 2.3). In the following protocol, parameter ω controls the number of parallel executions of randomized OTs. The protocol's security parameter is composed of ω and the underlying cryptosystem's security parameter n. The protocol has 10 rounds and communication complexity in the order of $O(\omega n)$. The exact communication complexity depends on the relation between ω and n, which in turn depends on the desired security level and the hardness of solving Alekhnovich's LPN problem with the currently best attack.

Protocol 2
Inputs: The sender $\mathbf{S}$ takes as input two bits x_0 and x_1, while the receiver $\mathbf{R}$ takes as input a choice bit c.
Common reference string: A random public key ck used for the commitment scheme.

1. Upon being activated with their inputs, the parties query $\mathcal{F}_{CRS}$ with $(\mathtt{sid}, \mathbf{S}, \mathbf{R})$ and receive $(\mathtt{sid}, \mathsf{crs})$ as answer.
2. $\mathbf{R}$ initiates the first round by performing the following actions:
   (a) $\mathbf{R}$ initially samples a random bit string $d \leftarrow \{0,1\}^\omega$, where, d_i denotes each bit in d for $i = 1, \ldots, \omega$.
   (b) For $i = 1, \ldots, \omega$, $\mathbf{R}$ generates a public-key pk_i and a secret-key sk_i, and sets $K_{i,d_i} = \mathsf{pk}_i = (A_i, B_i, G_i)$.
   (c) $\mathbf{R}$ commits to all public keys K_{i,d_i} by sending to $\mathbf{S}$ the message $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(K_{1,d_1}), \ldots, \mathsf{Com}_{\mathsf{ck}}(K_{\omega,d_\omega}))$.
3. Both parties run a coin tossing protocol in order to obtain random matrices:
   (a) $\mathbf{S}$ samples uniformly random matrices of the same size as the public key matrices $A'_i \in \mathbb{F}_2^{\ell_1 \times n_1}$, $B'_i \in \mathbb{F}_2^{\ell_2 \times n_1}$, $G'_i \in \mathbb{F}_2^{\ell_2 \times n_1}$, assigns $R'_i = (A'_i, B'_i, G'_i)$ and sends a commitment $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(R'_1), \ldots, \mathsf{Com}_{\mathsf{ck}}(R'_\omega))$ to $\mathbf{R}$.
   (b) For $i = 1, \ldots, \omega$, $\mathbf{R}$ samples uniformly random $A''_i \in \mathbb{F}_2^{\ell_1 \times n_1}$, $B''_i \in \mathbb{F}_2^{\ell_2 \times n_1}$, $G''_i \in \mathbb{F}_2^{\ell_2 \times n_1}$, assigns $R''_i = (A''_i, B''_i, G''_i)$ and sends $(\mathtt{sid}, R''_1, \ldots, R''_\omega)$ to $\mathbf{S}$.
   (c) $\mathbf{S}$ opens its commitments and for $i = 1, \ldots, \omega$ both parties compute $R_i = (\bar{A}_i = A'_i + A''_i, \bar{B}_i = B'_i + B''_i, \bar{C}_i = C'_i + C''_i)$.
4. $\mathbf{R}$ computes the remaining keys as follows:
   (a) For $i = 1, \ldots, \omega$, $\mathbf{R}$ sets $K_{i,\overline{d_i}} = K_{i,d_i} + R_i = (A_i + \bar{A}_i, B_i + \bar{B}_i, G_i + \bar{G}_i)$, scrambling the valid keys related to the random choice bit using the random matrices obtained in the coin tossing.
   (b) $\mathbf{R}$ sends all the resulting keys $K_{i,1} = (\tilde{A}_i, \tilde{B}_i, \tilde{G}_i)$ to $\mathbf{S}$ as $(\mathtt{sid}, K_{1,1}, \ldots, K_{\omega,1})$.
5. $\mathbf{S}$ computes $K_{i,0} = K_{i,1} + R_i = (\tilde{A}_i + \bar{A}_i, \tilde{B}_i + \bar{B}_i, \tilde{G}_i + \bar{G}_i)$ obtaining the public key pairs $K_{i,0}, K_{i,1}$, for $i = 1, \ldots, \omega$.
6. Both parties run a coin tossing protocol in order to obtain a random bit string Ω:
   (a) $\mathbf{S}$ samples a random bit string $v \leftarrow \{0,1\}^\omega$ and sends a commitment $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$ to $\mathbf{R}$.
   (b) $\mathbf{R}$ chooses a random bit string v' and sends $(\mathtt{sid}, v')$ to $\mathbf{S}$.
   (c) $\mathbf{S}$ opens its commitment and both parties compute $\Omega = v \oplus v'$.

7. Let I be the set of indexes $i \in \{1, \ldots, \omega\}$ such that $\Omega_i = 1$ and let J be the set of indexes $j \in \{1, \ldots, \omega\}$ such that $\Omega_j = 0$. **R** performs the following actions:
   - **Verification:** For each $i \in I$, **R** opens the commitments to K_{i,d_i} by sending $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(K_{i,d_i}))$.
   - **Derandomization:** For each $j \in J$, let ρ_j be a reordering bit such that if $\rho_j = 1$ the keys $K_{j,0}, K_{j,1}$ are swapped and if $\rho_j = 0$ they are left as they are. For each $j \in J$, **R** sends $(\mathtt{sid}, \rho_j)$ to **S** in such a way that, after the reordering, all the keys $K_{j,c}$ are valid.[5]
8. For each opening $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(K_{i,d_i}))$ that it receives, **S** checks that the public key pair $K_{i,0}, K_{i,1}$ is honestly generated (*i.e.* that there exists $b \in \{0,1\}$ s.t. $K_{i,b} = K_{i,d_i}$ and $K_{i,\overline{b}} = K_{i,d_i} \oplus R_i$). If this check fails for at least one public key pair **S** aborts, otherwise it continues as follows:
   - For each reordering bit ρ_j received by **S**, it derandomizes the corresponding public key pair by computing $(\hat{K}_{j,0}, \hat{K}_{j,1}) = K_{j,0\oplus\rho}, K_{j,1\oplus\rho}$.
   - Let μ be the number of indexes in J, and let $j_1, ..., j_\mu$ denote each of these indexes. For $j = j_1, ..., j_\mu$, **S** generates μ bits $x_{j,0}$ such that $x_{j_1,0} \oplus \cdots \oplus x_{j_\mu,0} = x_0$ and μ bits $x_{j,1}$ such that $x_{j_1,1} \oplus \cdots \oplus x_{j_\mu,1} = x_1$.
   - For $j = j_1, ..., j_\mu$, **S** encrypts $x_{j,0}$ under public key $\hat{K}_{j,0}$ and encrypts $x_{j,1}$ under public key $\hat{K}_{j,1}$ by computing $\mathsf{ct}_{j,0} = \mathsf{Enc}(\hat{K}_{j,0}, x_{j,0})$ and $\mathsf{ct}_{j,1} = \mathsf{Enc}(\hat{K}_{j,1}, x_{j,1})$.
   - **S** sends all ciphertexts to **R** as $(\mathtt{sid}, (\mathsf{ct}_{j_1,0}, \mathsf{ct}_{j_1,1}), \ldots, (\mathsf{ct}_{j_\mu,0}, \mathsf{ct}_{j_\mu,1}))$.
9. For $j = j_1, ..., j_\mu$, **R** decrypts the ciphertexts related to x_c by computing $x_{j,c} = \mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_{j,c})$. If any of the decryption attempts fail, **R** outputs a random $x_c \leftarrow \{0,1\}$. Otherwise, **R** outputs $x_c = x_{j_1,c} \oplus \ldots \oplus x_{j_\mu,c}$.

Correctness It is clear that the protocol runs in polynomial time. The classical coin tossing protocol ensures that the string Ω and matrices R_i are uniformly distributed and the commitment hiding property ensures that **S** cannot obtain any information about the keys in the unopened commitments.

Notice that, after the reordering, all the public key pairs $(\hat{K}_{j,0}, \hat{K}_{j,1})$ are such that $\hat{K}_{j,c}$ is a valid public key and $\hat{K}_{j,\overline{c}}$ is a scrambled public key (*i.e.* summed with the random matrices in R_j). Thus, **R** is able to decrypt all of the ciphertexts $\mathsf{ct}_{j,c}$ for $j = j_1, ..., j_\mu$, obtaining all bits $x_{j,c}$ that are necessary to compute the bit $x_c = x_{j_1,c} \oplus \ldots \oplus x_{j_\mu,c}$. On the other hand, **R** cannot obtain $x_{\overline{c}}$ through decrypting the ciphertexts $\mathsf{ct}_{i,\overline{c}}$, since they were generated under the scrambled keys. **S** cannot obtain the choice bit c by distinguishing the valid public keys from randomized keys, since the public-key of the cryptosystem is pseudorandom [31, 17].

Theorem 3. *Protocol 2 securely realizes the functionality $\mathcal{F}_{OT}$ in the $\mathcal{F}_{CRS}$-hybrid model under Assumption 1. Let π denote Protocol 2. For every PPT static malicious adversary $\mathcal{A}$ there is a PPT simulator $\mathcal{S}$ such that for all PPT environment $\mathcal{Z}$, the following holds:*

$$EXEC_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F}_{OT},\mathcal{S},\mathcal{Z}}$$

4.1 Security Proof

In this section we analyse the security of Protocol 2 by constructing a simulator $\mathcal{S}$ that interacts with $\mathcal{F}_{OT}$ such that no environment $\mathcal{Z}$ can distinguish between interactions with a static adversary $\mathcal{A}$ in the real world and interactions with $\mathcal{S}$ in the ideal world. The formal description of the simulator and the full proof of Theorem 3 showing that execution with $\mathcal{S}$ is indeed indistinguishable from execution with $\mathcal{A}$ are left for the full version of this paper. We first present trivial simulation cases (where both parties are honest or corrupted) and then consider the cases where only **S** or only **R** is corrupted separately. The simulators are based on techniques introduced in [35] and [15]. For each corruption scenario, $\mathcal{S}$

[5] If the operation performed with ρ is seen as computing $(\hat{K}_{j,0}, \hat{K}_{j,1}) = K_{j,0\oplus\rho}, K_{j,1\oplus\rho}$, the choice of ρ can be seen as $\rho = d_j \oplus c$. Here **R** makes sure that the public keys in the unopened commitments that will be used to encrypt the bit x_c (related to its choice bit) are valid public keys.

works as follows:

Simulating Communication with $\mathcal{Z}$: $\mathcal{S}$ writes all the messages received from $\mathcal{Z}$ in $\mathcal{A}$'s input tape, simulating $\mathcal{A}$'s environment. Also, $\mathcal{S}$ writes all messages from $\mathcal{A}$'s output tape to its own output tape, forwarding them to $\mathcal{Z}$.

Simulating trivial cases: If both **S** and **R** are corrupted, $\mathcal{S}$ simply runs $\mathcal{A}$ internally. Notice that $\mathcal{A}$ will generate the messages from both corrupted **S** and **R**. If neither **S** and **R** are corrupted, $\mathcal{S}$ runs the protocol between honest **S** and **R** internally on the inputs provided by $\mathcal{Z}$. All messages are delivered to $\mathcal{A}$.

Simulator for a Corrupted S If only **S** is corrupted, the simulator $\mathcal{S}$ has to extract the bits x_0 and x_1 (the adversary's input) by interacting with adversary $\mathcal{A}$ through Protocol 2. The main trick for doing this lies in cheating the coin tossing phase by means of the underlying commitment scheme's extractability. The simulator will use this ability to construct public key pairs where both keys are valid (allowing it to obtain both bits) and pass the corrupted **S**'s verification without getting caught. $\mathcal{S}$ sends the x_0 and x_1 obtained after decryption to $\mathcal{F}_{OT}$ and terminates. The simulator $\mathcal{S}$ is formally described as follows:

Simulating $\mathcal{F}_{CRS}$: $\mathcal{S}$ generates a commitment key $\mathsf{ck} \leftarrow \mathsf{Gen}(1^n)$ for which he knows the secret key tk and sets $\mathsf{crs} = \mathsf{ck}$. Later on, the secret key will be used as a trapdoor to extract unopened commitments. When the parties query $\mathcal{F}_{CRS}$, $\mathcal{S}$ hands them $(\mathtt{sid}, \mathsf{crs})$.

When the dummy **S** is activated, $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ initiates the first round by performing the following actions:
   (a) $\mathcal{S}$ initially samples a random bit string $d \leftarrow \{0,1\}^\omega$, where d_i denotes each bit in d for $i = 1, \ldots, \omega$.
   (b) For $i = 1, \ldots, \omega$, $\mathcal{S}$ generates a public-key pk_i and a secret-key sk_i, and sets $K_{i,d_i} = \mathsf{pk}_i = (A_i, B_i, G_i)$.
   (c) $\mathcal{S}$ commits to all public keys K_{i,d_i} by sending to $\mathcal{A}$ the message $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(K_{1,d_1}), \ldots, \mathsf{Com}_{\mathsf{ck}}(K_{\omega,d_\omega}))$.
2. $\mathcal{S}$ performs the coin tossing to generate the random matrices as follows:
   (a) Upon receiving $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(R'_1), \ldots, \mathsf{Com}_{\mathsf{ck}}(R'_\omega))$ from $\mathcal{A}$, $\mathcal{S}$ extracts the $R'_i = (A'_i, B'_i, G'_i)$.
   (b) $\mathcal{S}$ chooses public-keys $\mathsf{pk}_{i,\overline{d_i}} = (\overline{A}_i, \overline{B}_i, \overline{G}_i)$ with the respective secret-key, sets $K_{i,\overline{d_i}} = \mathsf{pk}_{i,\overline{d_i}}$ and computes $R''_i = R'_i \oplus \mathsf{pk}_{i,\overline{d_i}} = (\overline{A}_i + A'_i, \overline{B}_i + B'_i, \overline{G}_i + G'_i)$ for $i = 1, \ldots, \omega$. $\mathcal{S}$ sends $(\mathtt{sid}, R''_1, \ldots, R''_\omega)$ to $\mathcal{A}$.
3. Upon receiving the openings from $\mathcal{A}$, $\mathcal{S}$ sends $\mathsf{pk}_{1,1}, \ldots, \mathsf{pk}_{\omega,1}$ to $\mathcal{A}$.
4. $\mathcal{S}$ simulates the coin tossing:
   – Upon receiving $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$ from $\mathcal{A}$, $\mathcal{S}$ chooses a random bit string $v' \leftarrow \{0,1\}^\omega$ and sends to $\mathcal{A}$.
   – Upon receiving an opening $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(v))$ from $\mathcal{A}$, $\mathcal{S}$ computes $\Omega = v \oplus v'$ and stores $(\mathtt{sid}, \Omega)$. However, If $\mathcal{A}$ does not correctly open its commitment $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$, then $\mathcal{S}$ sends $\perp$ to $\mathcal{F}_{OT}$, simulating an invalid opening and halts.
5. After the coin tossing, $\mathcal{S}$ opens the commitments needed for verification and simulates reordering. Recall that i represents the indexes for which $\Omega_i = 1$ and j represents the indexes for which $\Omega_j = 0$.
   – **Verification:** For each i, $\mathcal{S}$ opens the commitments to K_{i,d_i} by sending $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(K_{i,d_i}))$.
   – **Derandomization:** For every j, $\mathcal{S}$ samples a random reordering bit $\rho_j \leftarrow \{0,1\}$. For each j, $\mathcal{S}$ sends $(\mathtt{sid}, \rho_j)$ to $\mathcal{A}$. [6]

[6] The reordering bit performs the same function described in the protocol for a honest receiver.

6. Upon receiving $(\texttt{sid}, (\mathsf{ct}_{j_1,0}, \mathsf{ct}_{j_1,1}), \ldots, (\mathsf{ct}_{j_\mu,0}, \mathsf{ct}_{j_\mu,1}))$, $\mathcal{S}$ uses the instructions of an honest receiver to decrypt and reconstruct both bits x_0 and x_1. For $j = j_1, ..., j_\mu$, $\mathcal{S}$ decrypts the ciphertexts related to x_{d_i} by computing $x_{j,d_i} = \mathsf{Dec}(sk_{j,d_i}, \mathsf{ct}_{j,d_i})$ and the ciphertexts related to $x_{\overline{d}_i}$ by computing $x_{j,\overline{d}_i} = \mathsf{Dec}(sk_{j,\overline{d}_i}, \mathsf{ct}_{j,\overline{d}_i})$ (notice that $\mathcal{S}$ knows all secret keys $sk_{j,d_i}, sk_{j,\overline{d}_i}$ since it cheated in the random padding generation). $\mathcal{S}$ obtains $x_{d_i} = x_{j_1,d_i} \oplus \ldots \oplus x_{j_\mu,d_i}$ and $x_{\overline{d}_i} = x_{j_1,\overline{d}_i} \oplus \ldots \oplus x_{j_\mu,\overline{d}_i}$. However, if $\mathcal{A}$ does not reply with a valid message or any of the decryption attempts fail, then $\mathcal{S}$ samples two random bits $x_0, x_1 \leftarrow \{0, 1\}$.
7. $\mathcal{S}$ completes the simulation by sending $(\texttt{sid}, \texttt{sender}, x_0, x_1)$ to $\mathcal{F}_{OT}$ as $\mathbf{S}$'s input and halts.

Lemma 1. *(Computational security for* $\mathbf{R}$*) Let π denote Protocol 2, for every PPT static malicious adversary $\mathcal{A}$ that corrupts only* $\mathbf{S}$ *and every PPT environment $\mathcal{Z}$, the following holds under Assumption 1:*

$$EXEC_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F}_{OT},\mathcal{S},\mathcal{Z}}$$

Proof. Simulator $\mathcal{S}$ only deviates from Protocol 2 in choosing the matrices R_i'', but this is indistinguishable from real protocol behavior since the public-keys of the cryptosystem are indistinguishable from random matrices of the same size. Later on, this will allow $\mathcal{S}$ to obtain both bits x_0 and x_1.

Notice that $\mathcal{A}$ opens its commitments with invalid values with at most negligible probability because of the commitment scheme's binding property. Hence, the coin tossing phase will succeed with high probability. The random reordering bits ρ_j sent by $\mathcal{S}$ are indistinguishable from real reordering bits sent in the protocol since the public keys are pseudorandom. Once again, $\mathcal{A}$ would have to distinguish the keys $K_{j,d_j}, K_{j,\overline{d}_j}$ from random matrices in order to detect that ρ_j is not generated as in the real protocol.

$\mathcal{S}$ is able to obtain both x_0 and x_1 since both public keys are valid in the pairs $K_{j,d_j}, K_{j,\overline{d}_j}$ used to encrypt $\mathcal{A}$'s messages and $\mathcal{S}$ knows the corresponding secret keys by cheating in the random paddings generation.

Simulator for a Corrupted R In this case where only $\mathbf{R}$ is corrupted, the simulator has to extract the choice bit c (the adversary's input) by interacting with the adversary $\mathcal{A}$ through Protocol 2. First, simulator $\mathcal{S}$ sets the CRS in such a way that it can extract the commitments sent by $\mathcal{A}$ in the first step. $\mathcal{S}$ runs the protocol as an honest $\mathbf{S}$, only deviating to extract the commitments containing the valid public key sent by $\mathcal{A}$. After the public key pairs are reordered, $\mathcal{S}$ verifies which key $\hat{K}_{j,0}$ or $\hat{K}_{j,1}$ corresponds to the valid public key K_{j,d_j} in the extracted (but unopened) commitment. The choice bit is determined as the bit c such that $\hat{K}_{j,c} = K_{j,d_j}$. $\mathcal{S}$ sends c to $\mathcal{F}_{OT}$, obtaining x_c in return. $\mathcal{S}$ then encrypts x_c and a dummy x_{1-c} using the procedure of a honest sender, sends the corresponding message to $\mathcal{A}$ and terminates. The simulator $\mathcal{S}$ is formally described as follows:

Simulating $\mathcal{F}_{CRS}$: $\mathcal{S}$ generates a commitment key $\mathsf{ck} \leftarrow \mathsf{Gen}(1^n)$ for which he knows the secret key tk and sets $\mathsf{crs} = \mathsf{ck}$. Later on, the secret key will be used as a trapdoor to extract unopened commitments. When the parties query $\mathcal{F}_{CRS}$, $\mathcal{S}$ hands them $(\texttt{sid}, \mathsf{crs})$.

When the dummy $\mathbf{R}$ is activated, $\mathcal{S}$ proceeds as follows:

1. Upon receiving $(\texttt{sid}, \mathsf{Com}_{\mathsf{ck}}(K_{1,d_1}), \ldots, \mathsf{Com}_{\mathsf{ck}}(K_{\omega,d_\omega}))$ from $\mathcal{A}$, $\mathcal{S}$ extract the commitments and stores $(\texttt{sid}, K_{1,d_1}, \ldots, K_{\omega,d_\omega})$.
2. $\mathcal{S}$ simulates the coin tossing to obtain random matrices as follows:
   (a) $\mathcal{S}$ samples uniformly random matrices of the same size as the public key matrices $A_i' \in \mathbb{F}_2^{\ell_1 \times n_1}$, $B_i' \in \mathbb{F}_2^{\ell_2 \times n_1}$, $G_i' \in \mathbb{F}_2^{\ell_2 \times n_1}$, assigns $R_i' = (A_i', B_i', G_i')$ and sends a commitment $(\texttt{sid}, \mathsf{Com}_{\mathsf{ck}}(R_1'), \ldots, \mathsf{Com}_{\mathsf{ck}}(R_\omega'))$ to to $\mathcal{A}$.
   (b) Upon receiving $(\texttt{sid}, R_1'', \ldots, R_\omega'')$ from $\mathcal{A}$, $\mathcal{S}$ opens its commitments and both parties compute $R_i = \left(\bar{A}_i = A_i' + A_i'', \bar{B}_i = B_i' + B_i'', \bar{C}_i = C_i' + C_i''\right)$ for $i = 1, \ldots, \omega$.

(c) Upon receiving $(\mathtt{sid}, K_{1,1}, \ldots, K_{\omega,1})$ for $K_{i,1} = (\tilde{A}_i, \tilde{B}_i, \tilde{G}_i)$ from $\mathcal{A}$, $\mathcal{S}$ computes $K_{i,0} = K_{i,1} + R_i = (\tilde{A}_i + \bar{A}_i, \tilde{B}_i + \bar{B}_i, \tilde{G}_i + \bar{G}_i)$ obtaining the public key pairs $K_{i,0}, K_{i,1}$, for $i = 1, \ldots, \omega$. .

3. Simulating the coin tossing phase:
   - $\mathcal{S}$ samples a random bit string $v \leftarrow \{0,1\}^\omega$ and sends a commitment $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$ to $\mathcal{A}$.
   - Upon receiving $\mathcal{A}$'s string $(\mathtt{sid}, v')$, $\mathcal{S}$ opens its commitment sending $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(v))$ to $\mathcal{A}$ and receives.
   - $\mathcal{S}$ computes $\Omega = v \oplus v'$.
4. Let i represent the indexes for which $\Omega_i = 1$ and j represent the indexes for which $\Omega_j = 0$. Upon receiving the openings and the reordering bits $(\mathtt{sid}, \rho_j)$ from $\mathcal{A}$, $\mathcal{S}$ proceeds as follows. If $\mathcal{A}$ send invalid openings, then $\mathcal{S}$ sends $\perp$ to $\hat{\mathcal{F}}_{OT}$, simulating an abortion and halts; otherwise do the following:
   - For each opening $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(K_{i,d_i}))$, s uses the key K_{i,d_i} and the instructions of an honest sender to check whether the public key pairs are valid (*i.e.* one of the keys is equal to K_{i,d_i} and the other is equal to $K_{i,d_i} \oplus R_i$). If this check fails, $\mathcal{S}$ sends $\perp$ to $\hat{\mathcal{F}}_{OT}$, simulating an abortion and halts. Otherwise it continues to the next step.
   - For each reordering bit ρ_j received by $\mathcal{S}$, it derandomizes the corresponding public key pair by computing $(\hat{K}_{j,0}, \hat{K}_{j,1}) = K_{j,0\oplus\rho}, K_{j,1\oplus\rho}$.
   - $\mathcal{S}$ uses the keys K_{j,d_j} obtained from the extracted commitments to find at least one valid reordered pair $(\hat{K}_{j,0}, \hat{K}_{j,1})$. If no such pair is found, $\mathcal{S}$ aborts, sending $\perp$ to $\hat{\mathcal{F}}_{OT}$ and halting. Otherwise, $\mathcal{S}$ obtains c by checking which key in the pair is equal to K_{j,d_j}, *i.e.* if $\hat{K}_{j,0} = K_{j,d_j}$ then $c = 0$ and if $\hat{K}_{j,1} = K_{j,d_j}$ then $c = 1$.
   - $\mathcal{S}$ sends $(\mathtt{sid}, \mathtt{receiver}, c)$ to $\hat{\mathcal{F}}_{OT}$, receiving $(\mathtt{sid}, x_c)$ in response.
5. $\mathcal{S}$ samples a random bit $x_{\overline{c}} \leftarrow \{0,1\}$, obtaining a pair (x_0, x_1) since it already learned x_c from $\hat{\mathcal{F}}_{OT}$. $\mathcal{S}$ completes the protocol by performing the following actions:
   - Let μ be the number of indexes j, and let $j_1, ..., j_\mu$ denote each of these indexes. For $j = j_1, ..., j_\mu$, $\mathcal{S}$ generates μ bits $x_{j,0}$ such that $x_{j_1,0} \oplus \cdots \oplus x_{j_\mu,0} = x_0$ and μ bits $x_{j,1}$ such that $x_{j_1,1} \oplus \cdots \oplus x_{j_\mu,1} = x_1$.
   - For $j = j_1, ..., j_\mu$, $\mathcal{S}$ encrypts $x_{j,0}$ under public key $\hat{K}_{j,0}$ and encrypts $x_{j,1}$ under public key $\hat{K}_{j,1}$ by computing $\mathsf{ct}_{j,0} = \mathsf{Enc}(\hat{K}_{j,0}, x_{j,0}; \mathsf{r}_{j,0})$ and $\mathsf{ct}_{j,1} = \mathsf{Enc}(\hat{K}_{j,1}, x_{j,1}; \mathsf{r}_{j,1})$, respectively.
   - $\mathcal{S}$ sends all ciphertexts to $\mathcal{A}$ as $(\mathtt{sid}, (\mathsf{ct}_{j_1,0}, \mathsf{ct}_{j_1,1}), \ldots, (\mathsf{ct}_{j_\mu,0}, \mathsf{ct}_{j_\mu,1}))$.

Lemma 2. *(Computational security for* **S***) Let π denote Protocol 2, for every PPT static malicious adversary $\mathcal{A}$ that corrupts only* **R** *and every PPT environment $\mathcal{Z}$, the following holds under Assumption 1:*

$$EXEC_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\hat{\mathcal{F}}_{OT},\mathcal{S},\mathcal{Z}}$$

Proof. The only differences in the behavior of $\mathcal{S}$ and a real honest **S** following the protocol lie in that $\mathcal{S}$ may halt if the public key pairs related to the unopened commitments are invalid and that $\mathcal{S}$ generates a random $x_{\overline{c}}$. In this proof we will show that this deviations from the real protocol are indistinguishable from a real execution.

First, notice that the simulator only halts in the coin tossing phase and in the verification phase if $\mathcal{A}$ is able to break the commitment scheme's binding property and send invalid openings, which $\mathcal{A}$ is able to do with at most negligible probability.

$\mathcal{S}$ may halt when extracting the choice bit c if all the reordered public key pairs $(\hat{K}_{j,0}, \hat{K}_{j,1})$ related to the unopened commitments are invalid (*i.e.* neither of the public keys corresponds to the unopened commitment). Let μ be the number of indexes j for which $\Omega_j = 0$. In order to show that all the reordered public key pairs $(\hat{K}_{j,0}, \hat{K}_{j,1})$ are invalid with at most negligible probability, we will first show that $\mu \geq \frac{\omega}{2} - \eta$ with overwhelming probability for any constant $\eta > 0$ and ω sufficiently large, where $|\Omega| = \omega$. Let X_i be Bernoulli trials that represent the outcome Ω_i of each bit of the coin tossing for $i = 1, \ldots, \omega$ and let $X = \sum_{i=1}^{\omega} X_i$. It is clear that the expected value of X is $E[X] = \frac{\omega}{2}$. For a deviation η in the expected value of X, the Chernoff bounds then implies that $\mu \geq \frac{\omega}{2} - \eta$ with overwhelming probability. But note that each invalid pair of keys is detected with probability $\frac{1}{2}$ during the commitment opening. Therefore the probability that $\frac{\omega}{2} - \eta$ or more reordered public key pairs $(\hat{K}_{j,0}, \hat{K}_{j,1})$ related to the unopened commitments are all invalid and still the protocol is not aborted due to the detection of some invalid pair is at most $(1/2)^{\frac{\omega}{2}-\eta}$. Hence $\mathcal{S}$ halts with at most negligible probability.

The remaining deviation in $\mathcal{S}$'s procedure consists in sampling a random $x_{\overline{c}} \leftarrow \{0,1\}$ in order to obtain a pair of bits (x_0, x_1) with respect to $(x_c, x_{\overline{c}})$. $\mathcal{S}$ then uses the uniformly distributed bit as an input to generate the ciphertexts in the last message from $\mathcal{S}$ to $\mathcal{A}$. On the other hand, in a real execution an honest $\mathbf{S}$ knows both messages x_0 and x_1 and generates ciphertexts with respect to them. However, since at least one of the public keys used to encrypt the bits $x_{j_1,\overline{d}} \oplus \cdots \oplus x_{j_\mu,\overline{d}} = x_{\overline{d}}$ is a random matrix [7], it is not possible to distinguish the ciphertexts $\mathsf{ct}_{j_1,\overline{d}}, \ldots, \mathsf{ct}_{j_\mu,\overline{d}}$ generated by $\mathcal{S}$ from the equivalent ciphertexts generated in the real execution without breaking Assumption 1. Hence, the last message sent by $\mathcal{S}$ is computationally indistinguishable from the equivalent message of a real execution.

5 Conclusion

We introduce the first stand alone and universally composable oblivious transfer protocols based on a LPN style assumption (*i.e.* Alekhnovich's Average-Nearest codeword assumption). These results improve over previous OT protocols based on coding theory by requiring weaker assumptions. Moreover, our protocols provide a new building block for complex LPN based cryptographic schemes, specially the secure computation protocol of [30]. We leave open the questions of obtaining OT protocols secure against adaptive adversaries under the LPN assumption and its variants, as well as improving on the efficiency of our constructions. Building other cryptographic schemes, such as universally composable commitments, based on these assumptions is also an interesting open problem.

References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.
2. Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307, Cambridge, Massachusetts, USA, October 11–14, 2003. IEEE Computer Society Press.
3. Michael Alekhnovich. More on average case vs approximation complexity. *Computational Complexity*, 20:755–786, 2011.
4. Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and spplications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557, Santa Barbara, CA, USA, August 20–24, 1989. Springer, Berlin, Germany.
5. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.
6. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.
7. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *Journal of Cryptology*, 19(2):135–167, April 2006.
8. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[7] This is ensured by the test during the choice bit extraction described before.

9. Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Germany.
10. Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, Berlin, Germany, March 15–17, 2009.
11. Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88, Nara, Japan, February 26 – March 1, 2013. Springer, Berlin, Germany.
12. Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 350–354, Santa Barbara, CA, USA, August 16–20, 1987. Springer, Berlin, Germany.
13. Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Berlin, Germany.
14. Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. Essentially optimal universally composable oblivious transfer. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC 08*, volume 5461 of *LNCS*, pages 318–335, Seoul, Korea, December 3–5, 2008. Springer, Berlin, Germany.
15. Bernardo Machado David, Anderson C. A. Nascimento, and Jörn Müller-Quade. Universally composable oblivious transfer from lossy encryption and the mceliece assumptions. In Adam Smith, editor, *ICITS 2012*, volume 7412 of *LNCS*, pages 80–99. Springer, 2012.
16. Nico Döttling, Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A cca2 secure variant of the mceliece cryptossystem. *Information Theory, IEEE Transactions on*, to appear.
17. Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 485–503, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.
18. Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 240–251, San Francisco, CA, USA, April 20–24, 2009. Springer, Berlin, Germany.
19. Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. Oblivious transfer based on the mceliece assumptions. In Reihaneh Safavi-Naini, editor, *ICITS 2008*, volume 5155 of *LNCS*, pages 107–117, Calgary, Canada, August 10–13, 2008. Springer, Berlin, Germany.
20. Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. Oblivious transfer based on the mceliece assumptions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E95-A(2):567–575, 2012.
21. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.
22. David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295, Paris, France, May 26–28, 2010. Springer, Berlin, Germany.
23. Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 297–316, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.
24. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City,, New York, USA, May 25–27, 1987. ACM Press.
25. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
26. Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
27. Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012.

28. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany.
29. Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 99–108, Seattle, Washington, USA, May 21–23, 2006. ACM Press.
30. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
31. Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.
32. Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 97–114, Barcelona, Spain, May 20–24, 2007. Springer, Berlin, Germany.
33. Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.
34. Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.
35. Andrew Y. Lindell. Efficient fully-simulatable oblivious transfer. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 52–70, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.
36. Helger Lipmaa. New communication-efficient oblivious transfer protocols based on pairings. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008*, volume 5222 of *LNCS*, pages 441–454, Taipei, Taiwan, September 15–18, 2008. Springer, Berlin, Germany.
37. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN Progress Report 4244, Jet Propulsion Laboratory, 1978.
38. Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, New York, NY, USA, 2005.
39. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457, Washington, DC, USA, January 7–9, 2001. ACM-SIAM.
40. Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15:159–166, 1986.
41. Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the mceliece cryptosystem without random oracles. In *International Workshop on Coding and Cryptography (WCC)*, pages 257–268, 2007.
42. Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Des. Codes Cryptography*, 49(1-3):289–305, 2008.
43. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.
44. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.
45. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134, Santa Fe, New Mexico, November 20–22, 1994. IEEE Computer Society Press.
46. Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, January 1983.
47. Bingsheng Zhang, Helger Lipmaa, Cong Wang, and Kui Ren. Practical fully simulatable oblivious transfer with sublinear communication. In Ahmad-Reza Sadeghi, editor, *FC 2013*, volume 7859 of *LNCS*, pages 78–95, Okinawa, Japan, April 1–5, 2013. Springer, Berlin, Germany.

A Definitions

In this section we present basic definitions used throughout the paper.

A.1 Public-Key Encryption Schemes

A public-key encryption scheme (PKE) is defined as follows:

Definition 1 (Public-Key Encryption). *A public-key encryption scheme is a triplet of algorithms* Gen*,* Enc*,* Dec*.* Gen *is a probabilistic polynomial-time key generation algorithm which takes as input a security parameter* 1^n *and outputs a public-key* pk *and a secret-key* sk*. The public-key specifies the message space* $\mathcal{M}$ *and the ciphertext space* $\mathcal{C}$*.* Enc *is a (possibly) probabilistic polynomial-time encryption algorithm which receives as input a public-key* pk*, a message* $\mathsf{m} \in \mathcal{M}$ *and random coins* r*, and outputs a ciphertext* $\mathsf{ct} \in \mathcal{C}$*. We write* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$ *to indicate explicitly that the random coins* r *are used and* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$ *if fresh random coins are used.* Dec *is a deterministic polynomial-time decryption algorithm which takes as input a secret-key* sk *and a ciphertext* ct*, and outputs either a message* $\mathsf{m} \in \mathcal{M}$ *or an error symbol* $\perp$*. For any pair of public and secret-keys generated by* Gen *and any message* $\mathsf{m} \in \mathcal{M}$ *it holds that* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})) = \mathsf{m}$ *with overwhelming probability over the randomness used by* Gen *and the random coins* r *used by* Enc*.*

A standard security notion for public-key encryption is indistinguishability against chosen-plaintext attacks (IND-CPA) [25] which is defined below using a game approach as in [28].

Definition 2 (IND-CPA security). *To a two-stage adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against* PKE *we associate the following experiment.*

```
$\mathsf{Exp}^{\mathsf{CPA}}_{\mathsf{PKE},\mathcal{A}}(n)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$
$(\mathsf{m}_0, \mathsf{m}_1, state) \leftarrow \mathcal{A}_1(\mathsf{pk})$ s.t. $|\mathsf{m}_0| = |\mathsf{m}_1|$
$b \leftarrow \{0, 1\}$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$
$b' \leftarrow \mathcal{A}_2(\mathsf{ct}^*, state)$
If $b = b'$ return $1$, else return $0$.
```

We define the advantage of $\mathcal{A}$ *in the experiment as*

$$\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE},\mathcal{A}}(n) = \left| \Pr\left(\mathsf{Exp}^{\mathsf{CPA}}_{\mathsf{PKE},\mathcal{A}}(n) = 1\right) - \frac{1}{2} \right|$$

We say that PKE *is indistinguishable against chosen-plaintext attacks (IND-CPA) if for all probabilistic polynomial-time (PPT) adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *the advantage of* $\mathcal{A}$ *in the above experiment is a negligible function of* n*.*

A.2 Oblivious Transfer Protocols and Stand Alone Security

An oblivious transfer (OT) protocol is a protocol executed between two parties: the sender $\mathbf{S}$ and the receiver $\mathbf{R}$. $\mathbf{S}$ has two input bits, x_0 and x_1, and $\mathbf{R}$ has a choice bit c as input. In the end of the protocol $\mathbf{R}$ should learn x_c, but not $x_{\overline{c}}$; while $\mathbf{S}$ should learn nothing.

In the stand alone case, only one instance of the protocol is executed at a time. The stand alone security of an oblivious transfer protocol can be captured as follows (this is the definition of [33] adapted to protocols with more than two messages). Let n be the security parameter. Let $\mathbf{S}$ and $\mathbf{R}$ be probabilistic polynomial-time (PPT) Turing machines. Let $View_{\widetilde{\mathbf{S}}}(\widetilde{\mathbf{S}}(z), \mathbf{R}(c))$ and $View_{\widetilde{\mathbf{R}}}(\mathbf{S}(x_0, x_1), \widetilde{\mathbf{R}}(z))$ be the *views* of dishonest $\mathbf{S}$ and $\mathbf{R}$, respectively, which contains their inputs z, all local computations, and messages exchanged.

Definition 3 (Security of OT Protocols). *A protocol* $[\mathbf{S}, \mathbf{R}](x_0, x_1; c)$ securely implements oblivious transfer*, if its execution satisfy the following properties.*

Completeness: *When both parties are honest,* $\mathbf{R}$ *outputs* x_c *while* $\mathbf{S}$ *has no output.*

Security for $\mathbf{S}$***:*** *For every PPT adversary* $\tilde{\mathbf{R}}$*, every input* z*, and a (sufficiently long) random tape* r *chosen at random, there exists a choice bit* c *such that for* $x_c \in \{0,1\}$ *the distribution (taken over* $\mathbf{S}$*'s randomness) of runs of* $\tilde{\mathbf{R}}(z)$ *using randomness* r *with* $\mathbf{S}$ *having input* x_c *and* $x_{\overline{c}} = 0$ *is computationally indistinguishable from the distribution of runs with* $\mathbf{S}$ *having input* x_c *and* $x_{\overline{c}} = 1$*.*

Security for $\mathbf{R}$***:*** *For any PPT adversary* $\widetilde{\mathbf{S}}$*, any security parameter* n *and any input* z *of size polynomial in* n*, the view that* $\widetilde{\mathbf{S}}(z)$ *obtains when* $\mathbf{R}$ *inputs* $c = 0$ *is computationally indistinguishable from that of when* $\mathbf{R}$ *inputs* $c = 1$*, denoted:*

$$View_{\widetilde{\mathbf{S}}}(\widetilde{\mathbf{S}}(z), \mathbf{R}(0))|_z \overset{c}{\approx} View_{\widetilde{\mathbf{S}}}(\widetilde{\mathbf{S}}(z), \mathbf{R}(1))|_z.$$

Adversarial Models A protocol is said to be secure against honest-but-curious parties, if the previous definition holds in the case $\mathbf{S}$ and $\mathbf{R}$ follow the protocol. If the previous definitions hold even if one of the parties deviates from the protocol, it is said to be secure against malicious parties (or *active secure*).