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, Denmark
bernardo@cs.au.dk
[2] Institute of Theoretical Informatics,
Karlsruhe Institute of Technology, Germany
rafael.dowsley@kit.edu
[3] Department of Electrical Engineering,
University of Brasilia, Brazil
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) [42,40,21] 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 [24,31] 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

⋆ Supported by European Research Council Starting Grant 279447. The author acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research centre (supported by the Danish Strategic Research Council) within which part of this work was performed.

D. Gritzalis et al. (Eds.): CANS 2014, LNCS 8813, pp. 143–158, 2014.

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 [40,26], the Decisional Diffie-Hellman (DDH) assumption [4,35,1,43], the Quadratic Residuosity (QR) assumption [26], the N'th residuosity assumption [26], the hardness of the Subgroup Decision Problem [33], and the McEliece assumptions [19]. Since Shor's algorithm [41] 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 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 [34]. 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 [34] proposed a cryptosystem based on the hardness of the syndrome decoding problem. Later on, Niederreiter [36] 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 [37,38] 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 [2,3,28] 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) [28], message authentication codes (MAC) [28], pseudo random functions (PRFs) [28], signature schemes with constant overhead [28], zero-knowledge [29], and commitments [29].

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 [28]. 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 [39] that provides instantiations based on the DDH, QR and Learning With Errors (LWE) [39]. 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 [14,30], the DDH and the strong RSA assumptions [23] and the Decisional Composite Residuosity (DCR) assumption [30,11].

Another approach to obtain UC-secure oblivious transfer protocols is to take a stand alone secure OT protocol and use compilers [27,25,10] 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 [29], our results contribute towards obtaining more complex cryptographic protocols based on coding based assumptions weaker than McEliece. Unfortunately, the UC secure protocol we introduce is meant to demonstrate the feasibility of obtaining OT based on LPN and lacks on efficiency, having high round communication complexity. Addressing efficiency issues, as well as obtaining security against adaptive adversaries, is left as a future work.

1.3 Outline

In Section 2 we introduce the notation, assumptions and definitions used throughout the paper. In Section 3, we present the active secure universally composable OT protocol based on cut-and-choose techniques.

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, $\rho \in O(n^{-(1+2\epsilon)/(1-2\epsilon)})$, and $n_1, \ell_1, \ell_2 \in O(n^{2/(1-2\epsilon)})$. 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.[1]

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.

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

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 $\mathcal{F}_{CRS}^{\mathcal{D}}$ is taken from [9].

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

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

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

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

• Upon receiving a message $(\mathtt{sid}, \mathtt{sender}, x_0, x_1)$ 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 $(\mathtt{sid}, \mathtt{receiver}, c)$ from **R**, check if a $(\mathtt{sid}, \mathtt{sender}, \cdots)$ message was previously sent. If yes, send $(\mathtt{sid}, x_c)$ to **R**, sid to the adversary $\mathcal{S}$ and halt. If not, send nothing to **R** (but continue running).

Similarly to the framework of [39], our protocols reuse the same CRS for multiple oblivious transfer invocations. In order to achieve this, we employ the same techniques of UC with joint state (JUC) [9].

3 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. The stand alone protocol can be constructed from a IND-CPA secure cryptosystem following the paradigm of [4], previously employed in [19] to obtain OT based on the McEliece assumptions. Basically, the receiver $\mathbf{R}$ generates a valid public key, scrambles it with random matrices of the same size and sends both valid and scrambled keys to the sender $\mathbf{S}$. $\mathbf{S}$ encrypts each of its messages under one of the public keys provided by $\mathbf{R}$ and sends the ciphertexts back. $\mathbf{R}$ is able to decrypt only the ciphertext created with the valid public key, obtaining only one of the messages. On the other hand, $\mathbf{S}$ cannot distinguish between the valid and scrambled public keys generated by $\mathbf{R}$, thus not knowing which message $\mathbf{R}$ obtains.

In the universally composable protocol, $\mathbf{R}$ generates a number of valid public keys K_{i,d_i} for random d_i's and commits to them. Next, 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 $\mathbf{S}$, 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. 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. An extractable commitment scheme is employed, allowing the simulator to cheat and obtain the information necessary to carry out the simulation.

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 1
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) **R** commits to all public keys K_{i,d_i} by sending to **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) **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 **R**.
   (b) For $i = 1, \ldots, \omega$, **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 **S**.
   (c) **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 $i = 1, \ldots, \omega$.
4. **R** computes the remaining keys as follows:
   (a) For $i = 1, \ldots, \omega$, **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) **R** sends all the resulting keys $K_{i,1} = (\tilde{A}_i, \tilde{B}_i, \tilde{G}_i)$ to **S** as $(\mathtt{sid}, K_{1,1}, \ldots, K_{\omega,1})$.
5. **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) **S** samples a random bit string $v \leftarrow \{0,1\}^\omega$ and sends a commitment $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$ to **R**.
   (b) **R** chooses a random bit string v' and sends $(\mathtt{sid}, v')$ to **S**.
   (c) **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.[2]
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:

[2] 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.

- For each reordering bit ρ_j received by $\mathbf{S}$, 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 $j = j_1, ..., j_\mu$, $\mathbf{S}$ 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 $j = j_1, ..., j_\mu$, $\mathbf{S}$ 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 $\mathsf{ct}_{j,1} = \mathsf{Enc}(\hat{K}_{j,1}, x_{j,1})$.
- $\mathbf{S}$ sends all ciphertexts to $\mathbf{R}$ 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 $j = j_1, ..., j_\mu$, $\mathbf{R}$ 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, $\mathbf{R}$ outputs a random $x_c \leftarrow \{0,1\}$. Otherwise, $\mathbf{R}$ 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 $\mathbf{S}$ 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, $\mathbf{R}$ 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, $\mathbf{R}$ cannot obtain $x_{\overline{c}}$ through decrypting the ciphertexts $\mathsf{ct}_{i,\overline{c}}$, since they were generated under the scrambled keys. $\mathbf{S}$ cannot obtain the choice bit c by distinguishing the valid public keys from randomized keys, since the public-key of the cryptosystem is pseudorandom [29,17].

Theorem 1. *Protocol 1 securely realizes the functionality $\mathcal{F}_{OT}$ in the $\mathcal{F}_{CRS}$-hybrid model under Assumption 1. Let π denote Protocol 1. 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:*

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

3.1 Security Proof

In this section we analyse the security of Protocol 1 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 1 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 $\mathbf{S}$ or only $\mathbf{R}$ is corrupted separately. The simulators are

based on techniques introduced in [32] and [15]. For each corruption scenario, $\mathcal{S}$ 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 1. 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 in Appendix A.

Simulator for a Corrupted R. In this case where only **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 1. 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 **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 $\hat{K}_{j,d_j}$ in the extracted (but unopened) commitment. The choice bit is determined as the bit c such that $\hat{K}_{j,c} = \hat{K}_{j,d_j}$. $\mathcal{S}$ 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 in Appendix B.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced Oblivious Transfer: How to Sell Digital Goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS, Cambridge, Massachusetts, USA, October 11-14, pp. 298–307. IEEE Computer Society Press (2003)
3. Alekhnovich, M.: More on average case vs approximation complexity. Computational Complexity 20, 755–786 (2011)

4. Bellare, M., Micali, S.: Non-interactive oblivious transfer and spplications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 547–557. Springer, Heidelberg (1990)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, Las Vegas, Nevada, USA, October 14-17, pp. 136–145. IEEE Computer Society Press (2001)
6. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
7. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. Journal of Cryptology 19(2), 135–167 (2006)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, Montréal, Québec, Canada, May 19-21, pp. 494–503. ACM Press (2002)
9. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
10. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, black-box constructions of adaptively secure protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
11. Choi, S.G., Katz, J., Wee, H., Zhou, H.-S.: Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 73–88. Springer, Heidelberg (2013)
12. Crépeau, C.: Equivalence between two flavours of oblivious transfers. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 350–354. Springer, Heidelberg (1988)
13. Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)
14. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially optimal universally composable oblivious transfer. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 318–335. Springer, Heidelberg (2009)
15. David, B.M., Nascimento, A.C.A., Müller-Quade, J.: Universally composable oblivious transfer from lossy encryption and the mceliece assumptions. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 80–99. Springer, Heidelberg (2012)
16. Döttling, N., Dowsley, R., Müller-Quade, J., Nascimento, A.C.A.: A cca2 secure variant of the mceliece cryptossystem. IEEE Transactions on Information Theory (to appear)
17. Döttling, N., Müller-Quade, J., Nascimento, A.C.A.: IND-CCA secure cryptography based on a variant of the LPN problem. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 485–503. Springer, Heidelberg (2012)
18. Dowsley, R., Müller-Quade, J., Nascimento, A.C.A.: A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 240–251. Springer, Heidelberg (2009)
19. Dowsley, R., van de Graaf, J., Müller-Quade, J., Nascimento, A.C.A.: Oblivious transfer based on the mceliece assumptions. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 107–117. Springer, Heidelberg (2008)

20. Dowsley, R., van de Graaf, J., Müller-Quade, J., Nascimento, A.C.A.: Oblivious transfer based on the mceliece assumptions. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E95-A(2), 567–575 (2012)
21. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 205–210. Plenum Press, New York (1982)
22. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 279–295. Springer, Heidelberg (2010)
23. Garay, J.A.: Efficient and universally composable committed oblivious transfer and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)
24. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC, May 25-27, pp. 218–229. ACM Press, New York (1987)
25. Haitner, I.: Semi-honest to malicious oblivious transfer—the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
26. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. Journal of Cryptology 25(1), 158–193 (2012)
27. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) 38th ACM STOC, Seattle, Washington, USA, May 21-23, pp. 99–108. ACM (2006)
28. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, May 17-20, pp. 433–442. ACM Press (2008)
29. Jain, A., Krenn, S., Pietrzak, K., Tentes, A.: Commitments and efficient zero-knowledge proofs from learning parity with noise. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 663–680. Springer, Heidelberg (2012)
30. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
31. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC, Chicago, Illinois, USA, May 2-4, pp. 20–31. ACM Press (1988)
32. Lindell, A.Y.: Efficient fully-simulatable oblivious transfer. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
33. Lipmaa, H.: New communication-efficient oblivious transfer protocols based on pairings. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 441–454. Springer, Heidelberg (2008)
34. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Technical Report DSN Progress Report 4244, Jet Propulsion Laboratory (1978)
35. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA, Washington, DC, USA, January 7-9, pp. 448–457. ACM-SIAM (2001)
36. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory 15, 159–166 (1986)
37. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the mceliece cryptosystem without random oracles. In: International Workshop on Coding and Cryptography (WCC), pp. 257–268 (2007)

38. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the mceliece cryptosystem without random oracles. Des. Codes Cryptography 49(1-3), 289–305 (2008)
39. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
40. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University (1981)
41. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS, Santa Fe, New Mexico, November 20-22, pp. 124–134. IEEE Computer Society Press (1994)
42. Wiesner, S.: Conjugate coding. SIGACT News 15(1), 78–88 (1983)
43. Zhang, B., Lipmaa, H., Wang, C., Ren, K.: Practical fully simulatable oblivious transfer with sublinear communication. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 78–95. Springer, Heidelberg (2013)

A Simulator for a Corrupted S

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 (sid, 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 $(\mathsf{sid}, \mathsf{Com}_{\mathsf{ck}}(K_{1,d_i}), \ldots, \mathsf{Com}_{\mathsf{ck}}(K_{\omega,d_i}))$.
2. $\mathcal{S}$ performs the coin tossing to generate the random matrices as follows:
   (a) Upon receiving $(\mathsf{sid}, \mathsf{Com}_{\mathsf{ck}}(R'_1), \ldots, \mathsf{Com}_{\mathsf{ck}}(R'_\omega))$ 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 $(\mathsf{sid}, R''_1, \ldots, R''_\omega)$ 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 $(\mathsf{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$ from $\mathcal{A}$, $\mathcal{S}$ chooses a random bit string $v' \leftarrow \{0,1\}^\omega$ and sends to $\mathcal{A}$.
   – Upon receiving an opening $(\mathsf{sid}, \mathsf{Open}_{\mathsf{ck}}(v))$ from $\mathcal{A}$, $\mathcal{S}$ computes $\Omega = v \oplus v'$ and stores (sid, Ω). However, If $\mathcal{A}$ does not correctly open its commitment $(\mathsf{sid}, \mathsf{Com}_{\mathsf{ck}}(v))$, 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 $\mathcal{A}$. [3]
6. Upon receiving $(\mathtt{sid}, (\mathsf{ct}_{j_1,0}, \mathsf{ct}_{j_1,1}), \ldots, (\mathsf{ct}_{j_\mu,0}, \mathsf{ct}_{j_\mu,1}))$, $\mathcal{S}$ 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 $(\mathtt{sid}, \mathtt{sender}, x_0, x_1)$ to $\mathcal{F}_{OT}$ as $\mathbf{S}$'s input and halts.

B Simulator for a Corrupted R

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 (sid, crs).

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

1. Upon receiving $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(K_{1,d_i}, \ldots, \mathsf{Com}_{\mathsf{ck}}(K_{\omega,d_i})))$ from $\mathcal{A}$, $\mathcal{S}$ extract the commitments and stores $(\mathtt{sid}, K_{1,d_i}, \ldots, K_{\omega,d_i})$.
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 $(\mathtt{sid}, \mathsf{Com}_{\mathsf{ck}}(R'_1), \ldots, \mathsf{Com}_{\mathsf{ck}}(R'_\omega))$ to to $\mathcal{A}$.
   (b) Upon receiving $(\mathtt{sid}, R''_1, \ldots, R''_\omega)$ 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 $(\mathtt{sid}, K_{1,1}, \ldots, K_{\omega,1})$ from $\mathcal{A}$ where $K_{i,1} = (\tilde{A}_i, \tilde{B}_i, \tilde{G}_i)$, $\mathcal{S}$ 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:

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

- $\mathcal{S}$ 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 $(\mathtt{sid}, \mathsf{Open}_{\mathsf{ck}}(pk_i|sk_i))$ and reordering bits $(\mathtt{sid}, \rho_j)$ from $\mathcal{A}$, $\mathcal{S}$ performs the following actions. However, if $\mathcal{A}$ send invalid openings, then $\mathcal{S}$ sends $\perp$ to $\hat{\mathcal{F}}_{OT}$, simulating an abortion and halts.
   - 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 $K_{j,0} = K_{j,d_j}$ then $c = 0$ and if $K_{j,1} = K_{j,d_j}$ 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 $(\mathtt{sid}, (\mathsf{ct}_{j_1,0}, \mathsf{ct}_{j_1,1}), \ldots, (\mathsf{ct}_{j_\mu,0}, \mathsf{ct}_{j_\mu,1}))$.