A Framework for Efficient Adaptively Secure Composable Oblivious Transfer in the ROM

Paulo S. L. M. Barreto * Bernardo David† Rafael Dowsley‡
Kirill Morozov§ Anderson C. A. Nascimento ¶

Abstract

Oblivious Transfer (OT) is a fundamental cryptographic protocol that finds a number of applications, in particular, as an essential building block for two-party and multi-party computation. We construct a universally composable (UC) protocol for oblivious transfer secure against active adaptive adversaries from any OW-CPA secure public-key encryption scheme with certain properties in the random oracle model (ROM). In terms of computation, our protocol only requires the generation of a public/secret-key pair, six encryption operations and two decryption operation, apart from a few calls to the random oracle. In terms of communication, our protocol only requires the transfer of one public-key, four ciphertexts, six binary strings of size equal to the security parameter, and two binary strings of size equal to the OT's messages. Next, we show how to instantiate our construction under the low noise LPN, McEliece, QC-MDPC, and CDH assumptions. Our instantiations based on the low noise LPN, McEliece, and QC-MDPC assumptions are the first UC-secure OT protocols based on coding assumptions to achieve: 1) adaptive security, 2) low round complexity, 3) low communication and computational complexities. Previous results in this setting only achieved static security and used costly cut-and-choose techniques. Our CDH-based instantiation is the first UC-secure OT protocol based on this assumption.

1 Introduction

Oblivious transfer (OT) [72, 45] is one of the major protocols within the realm of modern cryptography. It is a fundamental building block for secure two-party and multi-party computation. In this work, we will mainly focus on 1-out-of-2 string oblivious transfer, which is a two-party protocol. Here, the sender (called Alice) inputs two strings m_0 and m_1, and the receiver (called Bob) inputs a choice bit c, and obtains m_c as the output. Bob must not be able to learn m_{1-c}, while Alice must not learn c. Since oblivious transfer is normally used within other protocols as a primitive, it is desirable to ensure that its security is guaranteed even under arbitrary composition, using the universal composability (UC) framework [13]. Given the possible development of full-scale quantum computers, it is natural to look for protocols that implement oblivious transfer based on assumptions that are not known to be broken by quantum adversaries.

*University of Washington - Tacoma

†Tokyo Institute of Technology. Email: bdavid@c.titech.ac.jp. This work was supported by the Input Output Cryptocurrency Collaborative Research Chair, which has received funding from Input Output HK.

‡Aarhus University. Email: rafael@cs.au.dk. This project has received funding from the European research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme (grant agreement No 669255). This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731583 (SODA).

§Tokyo Institute of Technology

¶University of Washington - Tacoma

1.1 Our contributions

We propose a framework for obtaining oblivious transfer, which is UC-secure against active adaptive adversaries in the random oracle model (ROM). At the high level, our construction works as follows. We use a public-key encryption (PKE) scheme satisfying the following three properties:

- *Property 1 (informal)*: Let the public-key space $\mathcal{PK}$ form a group with operation denoted by "$\star$". Then, for the public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$, such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$, where q is chosen uniformly at random from $\mathcal{PK}$, one cannot decrypt both ciphertexts encrypted using pk_0 and pk_1, respectively. In particular, when the public/secret-key pair $(\mathsf{pk}_c, \mathsf{sk}_c)$, $c \in \{0, 1\}$, is generated, the above relationship guarantees that pk_{1-c} that is chosen to satisfy the constraint $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$ is "substantially random", so that learning the messages encrypted with pk_{1-c} is hard.

- *Property 2 (informal):* pk obtained using the key generation algorithm is indistinguishable from a random element of $\mathcal{PK}$. Note that we assume in this work that, in general, not all the elements of $\mathcal{PK}$ may represent valid public-keys.

- *Property 3 (informal):* The PKE scheme must be "committing", meaning that it must be impossible to generate two pairs of randomness and plaintext messages $(\mathsf{r}_0, \mathsf{m}_0)$ and $(\mathsf{r}_1, \mathsf{m}_1)$ with $\mathsf{m}_0 \neq \mathsf{m}_1$ such that encrypting m_0 with randomness r_0 under a uniformly random public-key pk yields the same ciphertext as encrypting m_1 with randomness r_1 under the same public-key pk.

Now, in our construction, the receiver generates a key pair $(\mathsf{pk}_c, \mathsf{sk}_c)$, queries a random oracle with a random seed value s to obtain q, computes pk_{1-c} such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$, and sends pk_0 and s to the sender. The latter obtains pk_1, uses the public keys to encrypt seeds that are used to generate one-time pads (using the random oracle), which in turn she uses to encrypt her respective inputs, and sends the encryptions to the receiver. Intuitively, Property 2 now prevents the sender from learning the choice bit, while Property 1 ensures that the receiver learns at most one of the inputs. Property 3 is used in mechanism that is used to inform Alice that the protocol has finished.

Our general framework has the following interesting properties:

- It can be instantiated with several code-based assumptions, namely low noise LPN, McEliece, QC-MDPC, assumptions. When instantiated with the LPN or McEliece assumptions, our protocol is several orders of magnitude more efficient than previous construction that also achieve UC-security [29, 26].

- Our low noise LPN, McEliece and QC-MDPC based instantiations are the first adaptively secure universally composable OT protocols based on these coding assumptions.

- It can also be instantiated with the CDH assumption. Our framework provides, to the best of our knowledge, the first UC-secure construction of an oblivious transfer protocol based on the CDH assumption (Simplest OT [19] can only be proven assuming gap-DH groups). We achieve the same computational complexity of [70].

Concurrent Work: We have recently been made aware of other works designing adaptively secure universally composable OT protocol based on the DDH assumption [11, 50] in the ROM. Differently from our CDH-based construction, which is a corollary of a more general result, these

concurrent works are specific for a DDH-based construction. The original version of the work of Hauck and Loss [50] also suffers from an issue that affected a previous version of this work (and also the Simplest OT protocol [19]) and was pointed out by an anonymous reviewer. The authors of that work are currently working on a fix that will make the protocol secure under the DDH assumption (instead of CDH) [60]. We postpone a detailed complexity comparison with both of these works until the fixes to these protocols are made public. In comparison to the DDH-based protocol of [11], our CDH-based instantiation uses a weaker assumption and achieves the same computational complexity, at the small cost of one additional round and a bit bigger communication complexity.

1.2 Related Works

The idea of constructing OT using two public-keys — the "pre-computed" one and the "randomized" one dates back to the CDH-based protocol of Bellare and Micali [5]. It was proven secure in the stand-alone model, and required zero-knowledge proofs. Naor and Pinkas [65], in particular,[1] presented an improved and enhanced CDH-based protocol in the random oracle model under the same paradigm, however it was proven secure only in the half-simulation paradigm. It is worth noting that both of the above schemes are tailored for the Diffie-Hellman groups, and hence generalizing them is not trivial. Dowsley *et al.* [40] constructed oblivious transfer using the McEliece encryption and the group operation was bitwise exclusive-or of matrices (representing the public-keys). This construction required an expensive cut-and-choose technique, which was leveraged by David *et al.* [29] to show the UC-security of this construction. Although Mathew *et al.* [62] showed that the cut-and-choose techniques can be avoided in [40] without changing the assumptions, they only proved the stand-alone security of their proposal.

Most of the public-key cryptographic schemes that are deployed nowadays have their security based on hardness assumptions coming from number theory, such as factoring and computing discrete logarithms. Likewise, when it comes to (computationally secure) OT protocols in the UC-security setting, OT protocols can be designed under assumptions such as: Decisional Diffie-Hellman (DDH) [47, 70], strong RSA [47], Quadractic Residuosity [70], Decisional Linear (DLIN) [52, 24] and Decisional Composite Residuosity (DCR) [52, 18]. We would like to emphasize that although our CDH-based protocol is somewhat similar to the basic protocol by Naor and Pinkas [65], there are the following two crucial differences: 1) In their protocol, the sender chooses randomness that is used to "randomize" the receiver's keys, while in our protocol, that randomness comes from applying the random oracle to a seed chosen by the receiver himself; 2) Our protocol uses a different encryption method for the sender's messages, in particular, the ElGamal encryption is employed, which is not the case in their protocol. It is exactly those differences that allow us to leverage stronger security guarantees (UC-security) as compared to the Naor-Pinkas protocol (half-simulation security).

It has been known for more than two decades that Shor's algorithm [74] makes factoring and computing discrete logarithms easy for quantum computers. Therefore, an important pending problem concerns designing post-quantum OT protocols. One solution is to rely on statistically secure protocols (i.e., those not depending on any computational assumption and thus secure even if full-scale quantum computers become reality) based on assumptions such as the existence of noisy channels [21, 20, 22, 66, 71, 1, 39], pre-distributed correlated data [3, 73], cryptogates [57], the bounded storage model [12, 30, 34, 35] and on hardware tokens [32, 38]. Nonetheless, these constructions (except the ones based on a trusted initializer) are rather impractical. Thus,

[1] They have also presented a DDH-based OT protocol in the standard model, but we require the random oracle model as a setup assumption, and hence leave this scheme out of scope of our comparison.

it seems reasonable to focus on obtaining OT protocols based on computational problems that are believed to be hard even for quantum computers: such as, for instance, the Learning from Parity with Noise (LPN), the Learning with Errors (LWE), and the McEliece problems.

The LPN problem essentially states that given a system of binary linear equations, in which the outputs are disturbed by some noise, it is difficult to determine the solution. It is very simple to generate LPN samples, however finding the solution seems to be very hard [10, 61, 59]. Therefore it is an attractive assumption that has been widely used for symmetric cryptographic primitives [51, 53, 49, 56, 58]. It has seen far less usage in asymmetric cryptographic primitives. However, public-key encryption schemes [2, 25, 33] were designed based on the LPN variant introduced by Alekhnovich [2], which has low noise but only provides a linear amount of samples. An OT protocol was also designed based on this variant of LPN [26]. However, this protocol is based on cut-and-choose techniques, which require a number of key generation, encryption and decryption operations linear in the cut-and-choose statistical security parameter. On the other hand, the instantiation of our OT protocol using this LPN variant (Section 5.3) is far more efficient.

McEliece [63] introduced a public-key encryption scheme based on hardness of the syndrome decoding problem. The cryptosystem proposed by Niederreiter [67] is its dual. Later on, IND-CPA-secure [68, 69] and IND-CCA2-secure [37, 46, 31] variants of these schemes were introduced. Both stand-alone [41, 42, 62] as well as fully-simulatable [27, 28] and UC-secure [29] OT protocols can be built using these cryptosystems. As in the case of low noise LPN, most of these protocols (and all UC-secure ones) are based on cut-and-choose techniques, which require a number of key generation, encryption and decryption operations linear in the cut-and-choose statistical security parameter. At the same time, our construction uses a small constant number of these operations. All the above advanced constructions assume pseudorandomness of the public-keys of the McEliece and Niederreiter PKE's. Our OT protocol based on the McEliece encryption scheme (Section 5.1) is far more efficient than the ones using cut-and-choose techniques.

A number of adaptively secure universally composable oblivious transfer protocols have been proposed in the literature [15, 7, 9, 18, 48, 8, 17]. Most of these protocols are on the CRS model and use erasures in order to achieve adaptive security. Moreover, these protocols require more than 2 rounds and rely either on complex zero-knowledge proof techniques or on adaptively secure universally composable commitments as central building blocks of their constructions. On the other hand, our generic construction does not assume secure erasures and requires solely a simple OW-CPA secure cryptosystem and can be executed in 2 rounds, which is optimal. Moreover, by avoiding heavy primitives such as adaptively secure UC commitments and zero-knowledge proofs, we achieve much better concrete computational and communication complexities. Besides this stark contrast in efficiency, our results show that adaptively secure UC OT can be achieved under much weaker assumptions in the random oracle model in comparison to the CRS model.

It is a well-known fact that UC-secure OT protocols require some setup assumption [14]. Our protocols use the random oracle model as such. Alternative setup assumptions that can be used to obtain UC-secure OT protocols include the common reference string (CRS) model [15, 47, 70], the public-key infrastructure model [23], the existence of noisy channels [36, 43], and tamper-proof hardware [55, 32, 38].

2 Preliminaries

We denote by κ the security parameter. Let $y \xleftarrow{\$} F(x)$ denote running the randomized algorithm F with input x and random coins, and obtaining the output y. Similarly, $y \leftarrow F(x)$ is used

for a deterministic algorithm. For a set $\mathcal{X}$, let $x \xleftarrow{\$} \mathcal{X}$ denote x chosen uniformly at random from $\mathcal{X}$; and for a distribution $\mathcal{Y}$, let $y \xleftarrow{\$} \mathcal{Y}$ denote y sampled according to the distribution $\mathcal{Y}$. Let $\mathbb{F}_2$ denote the finite field with 2 elements. For $A, B \in \mathbb{F}_2^{m \times n}$, $A \oplus B$ denotes their element-wise exclusive-or. For a parameter ρ, χ_ρ denotes the Bernoulli distribution that outputs 1 with probability ρ. Let $\mathsf{HW}(e)$ denote the Hamming weight of a vector e, i.e., the number of its non-zero positions. We will denote by $\mathsf{negl}(\kappa)$ the set of negligible functions of κ. We abbreviate *probabilistic polynomial time* as PPT.

2.1 Encryption Schemes

The main building block used in our OT protocol is a public-key encryption scheme. Such a scheme, denoted as PKE, has public-key $\mathcal{PK}$, secret-key $\mathcal{SK}$, message $\mathcal{M}$, randomness $\mathcal{R}$ and ciphertext $\mathcal{C}$ spaces that are functions of the security parameter κ, and consists of the following three algorithms KG, Enc, Dec:

- The PPT key generation algorithm KG takes as input the security parameter 1^κ and outputs a pair of public $\mathsf{pk} \in \mathcal{PK}$ and secret $\mathsf{sk} \in \mathcal{SK}$ keys.

- The PPT encryption algorithm Enc takes as input a public-key $\mathsf{pk} \in \mathcal{PK}$, a message $\mathsf{m} \in \mathcal{M}$ and randomness $\mathsf{r} \in \mathcal{R}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$. We denote this operation by $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$. When r is not explcitly given as input, it is assumed to be sampled uniformly at random from $\mathcal{R}$.

- The (deterministic) decryption algorithm Dec takes as input the secret-key $\mathsf{sk} \in \mathcal{SK}$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ and outputs either a message $\mathsf{m} \in \mathcal{M}$ or an error symbol $\perp$. For $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$, any $\mathsf{m} \in \mathcal{M}$, and $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$, it should hold that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m}$ with overwhelming probability over the randomness used by the algorithms.

We should emphasize that for some encryption schemes not all $\widetilde{\mathsf{pk}} \in \mathcal{PK}$ are "valid" in the sense of being a possible output of KG. The same holds for $\widetilde{\mathsf{ct}} \in \mathcal{C}$ in relation to Enc and all possible coins and messages.

Next we define the notion of one-wayness against chosen-plaintext attacks (OW-CPA).

Definition 2.1 (OW-CPA security) *A* PKE *is OW-CPA secure if for every PPT adversary* $\mathcal{A}$*, and for* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$*,* $\mathsf{m} \xleftarrow{\$} \mathcal{M}$ *and* $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$*, it holds that*

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{ct}) = \mathsf{m}] \in \mathsf{negl}(\kappa).$$

Our OT framework uses as a building block a PKE that satisfies a variant of the OW-CPA security notion: informally, two random messages are encrypted under two different public-keys, one of which can be chosen by the adversary (but he does not have total control over both public-keys). His goal is then to recover both messages and this should be difficult. Formally, this property is captured by the following definition.

Property 2.2 *Consider the public-key encryption scheme* PKE *and the security parameter* κ*. It is assumed that* $\mathcal{PK}$ *forms a group with operation denoted by "*$\star$*". For every PPT two-stage adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *running the following experiment:*

$q \xleftarrow{\$} \mathcal{PK}$
$(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(q)$ *such that* $\mathsf{pk}_1, \mathsf{pk}_2 \in \mathcal{PK}$ *and* $\mathsf{pk}_1 \star \mathsf{pk}_2 = q$
$\mathsf{m}_i \xleftarrow{\$} \mathcal{M}$ *for* $i = 1, 2$
$\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_i)$ *for* $i = 1, 2$
$(\widetilde{\mathsf{m}_1}, \widetilde{\mathsf{m}_2}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{st})$

it holds that

$$\Pr[(\widetilde{\mathsf{m}_1}, \widetilde{\mathsf{m}_2}) = (\mathsf{m}_1, \mathsf{m}_2)] \in \mathsf{negl}(\kappa).$$

We also need a property about indistinguishability of a public-key generated using KG and an element sampled uniformly at random from $\mathcal{PK}$.

Property 2.3 *Consider the public-key encryption scheme* PKE *and the security parameter* κ. *Let* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ *and* $\mathsf{pk}' \xleftarrow{\$} \mathcal{PK}$. *For every PPT distinguisher* $\mathcal{A}$, *it holds that*

$$|\Pr[\mathcal{A}(\mathsf{pk}) = 1] - \Pr[\mathcal{A}(\mathsf{pk}') = 1]| \in \mathsf{negl}(\kappa).$$

Moreover, we need the PKE scheme to be committing, meaning that an adversary can only generate two different pairs of randomness and plaintext message that result in the same ciphertexts when encrypted under a uniformly random public-key with negligible probability.

Property 2.4 *Consider the public-key encryption scheme* PKE *and the security parameter* κ. *For every PPT adversary* $\mathcal{A}$, *it holds that:*

$$\Pr\left[\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_0, \mathsf{r}_0) = \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_1, \mathsf{r}_1) \;\middle|\; \begin{array}{c} \mathsf{pk} \xleftarrow{\$} \mathcal{PK}, \\ (\mathsf{r}_0, \mathsf{r}_1, \mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}(\mathsf{pk}), \\ \mathsf{r}_0 \in \mathcal{R},\ \mathsf{r}_1 \in \mathcal{R}, \\ \mathsf{m}_0 \neq \mathsf{m}_1,\ \mathsf{m}_0 \in \mathcal{M},\ \mathsf{m}_1 \in \mathcal{M} \end{array}\right] \in \mathsf{negl}(\kappa)$$

Note that if Properties 2.3 and 2.4 hold for some PKE, then the modified version of Property 2.4 in which pk is chosen using KG also trivially holds.

In Section 5, we prove that Properties 2.2, 2.3 and 2.4 hold for cryptosystems based on a number of standard assumptions, yielding different instantiations of our framework.

2.2 Universal Composability

We prove our protocols secure in the Universal Composability (UC) framework introduced by Canetti in [13]. In this section, we present a brief description of the UC framework originally given in [16] and refer interested readers to [13] for further details. In this framework, protocol security is analyzed under the real-world/ideal-world paradigm, *i.e.*, by comparing the real world execution of a protocol with an ideal world interaction with the primitive that it implements. The model includes a *composition theorem*, that basically states that UC secure protocols can be arbitrarily composed with each other without any security compromises. This desirable property not only allows UC secure protocols to effectively serve as building blocks for complex applications but also guarantees security in practical environments, where several protocols (or individual instances of protocols) are executed in parallel, such as the Internet.

In the UC framework, the entities involved in both the real and ideal world executions are modeled as PPT Interactive Turing Machines (ITM) that receive and deliver messages through their input and output tapes, respectively. In the ideal world execution, dummy parties (possibly

controlled by an ideal adversary $\mathcal{S}$ referred to as the *simulator*) interact directly with the ideal functionality $\mathcal{F}$, which works as a trusted third party that computes the desired primitive. In the real world execution, several 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 and ideal executions are controlled by the *environment* $\mathcal{Z}$, an entity that delivers inputs and reads the outputs of the individual parties, the adversary $\mathcal{A}$ and the simulator $\mathcal{S}$. After a real or ideal execution, $\mathcal{Z}$ outputs a bit, which is considered as the output of the execution. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ (that represents everything that happens outside of the protocol execution) is not able to efficiently distinguish between the real and ideal executions, thus implying that the real world protocol is as secure as the ideal functionality.

We denote by $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \bar{r})$ the output of the environment $\mathcal{Z}$ in the real-world execution of a protocol π between n parties with an adversary $\mathcal{A}$ under security parameter κ, input z and randomness $\bar{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_{P_1}, \ldots, r_{P_n})$, where $(z, r_{\mathcal{Z}})$, $r_{\mathcal{A}}$ and r_{P_i} are respectively related to $\mathcal{Z}$, $\mathcal{A}$ and party i. Analogously, we denote by $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z, \bar{r})$ the output of the environment in the ideal interaction between the simulator $\mathcal{S}$ and the ideal functionality $\mathcal{F}$ under security parameter κ, input z and randomness $\bar{r} = (r_{\mathcal{Z}}, r_{\mathcal{S}}, r_{\mathcal{F}})$, where $(z, r_{\mathcal{Z}})$, $r_{\mathcal{S}}$ and $r_{\mathcal{F}}$ are respectively related to $\mathcal{Z}$, $\mathcal{S}$ and $\mathcal{F}$. The real world execution and the ideal executions are respectively represented by the ensembles $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}} = \{\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \bar{r})\}_{\kappa\in\mathbb{N}}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = \{\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z, \bar{r})\}_{\kappa\in\mathbb{N}}$ with $z \in \{0,1\}^*$ and a uniformly chosen $\bar{r}$.

In addition to these two models of computation, the UC framework also considers the $\mathcal{G}$-hybrid world, where the computation proceeds as in the real-world with the additional assumption that the parties have access to an auxiliary ideal functionality $\mathcal{G}$. In this model, honest parties do not communicate with the ideal functionality directly, but instead the adversary delivers all the messages to and from the ideal functionality. We consider the communication channels to be ideally authenticated, so that the adversary may read but not modify these messages. Unlike messages exchanged between parties, which can be read by the adversary, the messages exchanged between parties and the ideal functionality are divided into a *public header* and a *private header*. The public header can be read by the adversary and contains non-sensitive information (such as session identifiers, type of message, sender and receiver). On the other hand, the private header cannot be read by the adversary and contains information such as the parties' private inputs. We denote the ensemble of environment outputs that represents an execution of a protocol π in a $\mathcal{G}$-hybrid model as $\mathsf{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$ (defined analogously to $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$). UC security is then formally defined as:

Definition 2.5 *An n-party ($n \in \mathbb{N}$) protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that, for every environment $\mathcal{Z}$, the following relation holds:*

$$\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \approx \mathsf{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$$

We say that a protocol is *statistically secure*, if the same holds for all $\mathcal{Z}$ with unbounded computing power.

2.2.1 Adversarial Model:

We consider a malicious adversary, which can deviate from the prescribed protocol in an arbitrary way. We call the adversary *static*, if he has to corrupt parties before execution starts and the corrupted (or honest) parties remain as such throughout the execution. We call the adversary *adaptive*, if he is able corrupt parties at any point in the protocol execution, and even after it.

2.2.2 Oblivious Transfer Ideal Functionality:

The basic 1-out-of-2 string oblivious transfer functionality $\mathcal{F}_{\text{OT}}$ as defined in [15] is presented in Figure 1.

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

$\mathcal{F}_{\text{OT}}$ interacts with a sender Alice and a receiver Bob. The lengths of the strings λ is fixed and known to both parties. $\mathcal{F}_{\text{OT}}$ proceeds as follows:

- Upon receiving a message $(\mathsf{sender}, sid, \vec{x}_0, \vec{x}_1)$ from Alice, where each $\vec{x}_i \in \{0,1\}^\lambda$, store the tuple $(sid, \vec{x}_0, \vec{x}_1)$. Ignore further messages from Alice with the same sid.
- Upon receiving a message $(\mathsf{receiver}, sid, c)$ from Bob, where $c \in \{0,1\}$, check if a tuple $(sid, \vec{x}_0, \vec{x}_1)$ was recorded. If yes, send $(\mathsf{received}, sid, \vec{x}_c)$ to Bob, send a delayed output $(\mathsf{received}, sid)$ to Alice and halt. Otherwise, send nothing to Bob, but continue running.

Figure 1: Functionality $\mathcal{F}_{\text{OT}}$.

2.2.3 Setup Assumptions:

Our constructions rely on the random oracle model [6], which can be modelled in the UC framework as the $\mathcal{F}_{\text{RO}}$-hybrid model. The random oracle functionality $\mathcal{F}_{\text{RO}}$ is presented in Figure 2. Our construction will actually use two instances of $\mathcal{F}_{\text{RO}}$: $\mathcal{F}_{\text{RO1}}$ with range $\mathcal{PK}$ and $\mathcal{F}_{\text{RO2}}$ with range $\{0,1\}^\lambda$.

Functionality $\mathcal{F}_{\text{RO}}$

$\mathcal{F}_{\text{RO}}$ is parameterized by a range $\mathcal{D}$. $\mathcal{F}_{\text{RO}}$ keeps a list L of pairs of values, which is initially empty, and proceeds as follows:

- Upon receiving a value (sid, m) from a party P_i or from $\mathcal{S}$, if there is a pair $(m, \hat{h})$ in the list L, set $h = \hat{h}$. Otherwise, choose $h \xleftarrow{\$} \mathcal{D}$ and store the pair (m, h) in L. Reply to the activating machine with (sid, h).

Figure 2: Functionality $\mathcal{F}_{\text{RO}}$.

3 The Framework

In this section, we introduce our 1-out-of-2 OT protocol and prove its UC security against static malicious adversaries in the $\mathcal{F}_{\text{RO}}$-hybrid model (*i.e.*, the random oracle model). Our protocol uses as a building block a public-key encryption scheme that satisfies Properties 2.2, 2.3 and and 2.4 (defined in Section 2.1). The basic high-level idea is that Bob picks two public-keys $\mathsf{pk}_0, \mathsf{pk}_1$ such that he only knows the secret key corresponding to pk_c (where c is his choice bit) and hands them to Alice. She then uses the two public-keys to transmit the messages in an encrypted way, so that Bob can only recover the message for which he knows the secret-key sk_c.

A crucial point in such schemes is making sure that Bob is only able to decrypt one of the messages. In order to enforce this property, our protocol relies on Property 2.2 and uses the random oracle to force the element q to be chosen uniformly at random from $\mathcal{PK}$. After generating the pair of secret and public key $(\mathsf{sk}_c, \mathsf{pk}_c)$, Bob samples a seed s, queries the random oracle $\mathcal{F}_{\text{RO1}}$ to obtain q and computes pk_{1-c} such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$. Bob then hands the

public-key pk_0 and the seed s to Alice, enabling her to also compute pk_1. Since the public-keys are indistinguishable according to Property 2.3, Alice learns nothing about Bob's choice bit. Next, Alice picks two uniformly random strings p_0, p_1, queries them to the random oracle $\mathcal{F}_{\text{RO2}}$ obtaining p'_0, p'_1 as response, and computes one-time pad encryptions of her messages m_0, m_1 as $m'_0 = m_0 \oplus p'_0$ and $m'_1 = m_1 \oplus p'_1$. Alice also encrypts p_0 and p_1 under pk_0 and pk_1, respectively, to obtain ct_0 and ct_1. Alice sends $(m'_1, m'_2, \mathsf{ct}_0, \mathsf{ct}_1)$ to Bob. Bob can use sk_c to decrypt ct_c obtaining p_c. He then queries p_c to the random oracle $\mathcal{F}_{\text{RO2}}$ obtaining p'_c as response, and retrieves $m_c = m'_c \oplus p'_c$. Due to Property 2.2, Bob will not be able to recover p_{1-c} in order to query it to the random oracle and to decrypt m'_{1-c}. Therefore, the security for Alice is also guaranteed.

In order to prove our protocol secure against a malicious receiver, we must construct a simulator that is able to extract the choice bit by communicating with an internal copy of an adversary who corrupts the receiver. In order to do so, the simulator relies on the fact that the adversary will query $\mathcal{F}_{\text{RO2}}$ on p_c, which allows it to learn c, obtain m_c from $\mathcal{F}_{\text{OT}}$ and simulate an answer from $\mathcal{F}_{\text{RO2}}$ that results in the adversary obtaining m_c. However, the basic protocol sketched above does not force the adversary to query $\mathcal{F}_{\text{RO2}}$ immediately after receiving the second round message. In this case, in the real world execution, an honest Alice would halt after sending its last message, believing the protocol is complete. On the other hand, in the ideal world, an honest Alice would only perceive the protocol as completed (and halt) after receiving $(\mathsf{received}, sid)$ from $\mathcal{F}_{\text{OT}}$, which only happens after the simulator sends $(\mathsf{receiver}, sid, c)$ to $\mathcal{F}_{\text{OT}}$. In order to avoid this discrepancy, we need to force the receiver to query the random oracle in such a way that the simulator can extract c while proving to Alice that this query was made, after which the sender can safely assume the protocol is completed and halt. We implement such a mechanism through steps 2(c), 3(b) and 4 of Protocol π_{OT}. Basically, Alice also fixes a challenge ch and sends messages under both pk_0 and pk_1 such that decrypting any of them allows Bob to recover the challenge. Due to the committing property (Property 2.4), it is possible to design the mechanism so that Bob can verify that the challenge he recovers is the same no matter which secret-key he uses, thus avoiding selective failure attacks. Instead of halting after sending the message in the second round, Alice waits for Bob to send a response matching the challenge, which Bob can only compute by querying $\mathcal{F}_{\text{RO3}}$ in a way that allows the simulator to extract c.

Our protocol π_{OT} is described in Figure 3. It can be instantiated using different public-key encryption schemes as described in Section 5.

3.1 Security Analysis

We now formally state the security of π_{OT}.

Theorem 3.1 *Let* PKE *be a public-key encryption scheme that satisfies Properties 2.2, 2.3 and 2.4. When instantiated with* PKE*, the Protocol π_{OT} UC-realizes the functionality $\mathcal{F}_{\text{OT}}$ against static adversaries in the $\mathcal{F}_{\text{RO}}$-hybrid model.*

Proof: In order to prove the security of π_{OT}, we must construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between interactions with an adversary $\mathcal{A}$ in the real world and with $\mathcal{S}$ in the ideal world. For the sake of clarity, we will describe the simulator $\mathcal{S}$ separately for the case where only Bob is corrupted and the case where only Alice is corrupted. In both cases, $\mathcal{S}$ 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 $\mathcal{Z}$. Notice that simulating the cases where both Alice and Bob are honest or corrupted is trivial. If

Protocol π_{OT}

Let PKE be a public-key encryption scheme that satisfies Properties 2.2, 2.3 and 2.4, and κ be the security parameter. Protocol π_{OT} is executed between Alice with inputs $m_0, m_1 \in \{0,1\}^\lambda$ and Bob with input $c \in \{0,1\}$. They interact with each other and with four instances of the random oracle ideal functionality $\mathcal{F}_{\mathrm{RO}}$ ($\mathcal{F}_{\mathrm{RO1}}$ with range $\mathcal{PK}$, $\mathcal{F}_{\mathrm{RO2}}$ with range $\{0,1\}^\lambda$, $\mathcal{F}_{\mathrm{RO3}}$ with range[a] $\{0,1\}^{\kappa+|\mathcal{R}|}$ and $\mathcal{F}_{\mathrm{RO4}}$ with range $\{0,1\}^\kappa$), proceeding as follows:

1. Bob generates a pair of keys $(\mathsf{pk}_c, \mathsf{sk}_c) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$. He samples a random string $s \xleftarrow{\$} \{0,1\}^\kappa$ and sends (sid, s) to $\mathcal{F}_{\mathrm{RO1}}$, obtaining (sid, q) as answer. Bob computes pk_{1-c} such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$ and sends (sid, s, pk_0) to Alice.
2. Upon receiving (sid, s, pk_0) from Bob, Alice proceeds as follows:
   (a) Queries $\mathcal{F}_{\mathrm{RO1}}$ with (sid, s), obtaining (sid, q) in response. Alice computes pk_1 such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$.
   (b) Samples $p_0, p_1 \xleftarrow{\$} \{0,1\}^\kappa$ and queries $\mathcal{F}_{\mathrm{RO2}}$ with (sid, p_0) and (sid, p_1), obtaining (sid, p'_0) and (sid, p'_1) as answers. Alice computes $m'_0 = p'_0 \oplus m_0$, $m'_1 = p'_1 \oplus m_1$, $\mathsf{ct}_0 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_0, p_0)$ and $\mathsf{ct}_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, p_1)$. Alice sends $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ to Bob.
   (c) Samples $w_0, w_1 \xleftarrow{\$} \{0,1\}^\kappa$ and queries $\mathcal{F}_{\mathrm{RO3}}$ with (sid, w_0) and (sid, w_1), obtaining (sid, w'_0) and (sid, w'_1) as answers. Alice samples $\mathsf{r}'_0, \mathsf{r}'_1 \xleftarrow{\$} \mathcal{R}$ and queries $(sid, w_0|w_1|\mathsf{r}'_0|\mathsf{r}'_1)$ to $\mathcal{F}_{\mathrm{RO4}}$ to obtain $ch \in \{0,1\}^\kappa$. Alice computes[a] $u_0 = w'_0 \oplus (w_1|\mathsf{r}'_1)$, $u_1 = w'_1 \oplus (w_0|\mathsf{r}'_0)$, $\mathsf{ct}'_0 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_0, w_0, \mathsf{r}'_0)$ and $\mathsf{ct}'_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, w_1, \mathsf{r}'_1)$. Alice sends $(sid, (u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1))$ to Bob[b].
3. Bob proceeds as follows:
   (a) Upon receiving $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ from Alice, Bob computes $p_c \leftarrow \mathsf{Dec}(\mathsf{sk}_c, \mathsf{ct}_c)$. If the decryption fails, he samples $m_c \xleftarrow{\$} \{0,1\}^\lambda$. Otherwise, he queries $\mathcal{F}_{\mathrm{RO2}}$ with (sid, p_c), obtaining (sid, p'_c) as answer. Bob computes $m_c = m'_c \oplus p'_c$
   (b) Upon receiving $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$ from Alice, Bob computes $w_c \leftarrow \mathsf{Dec}(\mathsf{sk}_c, \mathsf{ct}'_c)$. If decryption succeeds, he queries $\mathcal{F}_{\mathrm{RO3}}$ with (sid, w_c), obtaining (sid, w'_c) as answer. Bob computes[a] $(\widetilde{w}_{1-c}|\widetilde{\mathsf{r}}_{1-c}) = u_c \oplus w'_c$ and checks that $\mathsf{ct}'_{1-c} = \mathsf{Enc}(\mathsf{pk}_{1-c}, \widetilde{w}_{1-c}, \widetilde{\mathsf{r}}_{1-c})$. Bob queries $\mathcal{F}_{\mathrm{RO3}}$ with $(sid, \widetilde{w}_{1-c})$, obtaining $(sid, \widetilde{w}'_{1-c})$ as answer. Bob computes[a] $(\widetilde{w}_c|\widetilde{\mathsf{r}}_c) = u_{1-c} \oplus \widetilde{w}'_{1-c}$ and checks that $\mathsf{ct}'_c = \mathsf{Enc}(\mathsf{pk}_c, \widetilde{w}_c, \widetilde{\mathsf{r}}_c)$. If all tests are successful, then he queries $(sid, \widetilde{w}_0|\widetilde{w}_1|\widetilde{\mathsf{r}}_0|\widetilde{\mathsf{r}}_1)$ to $\mathcal{F}_{\mathrm{RO4}}$ to obtain $\widetilde{ch}$ and sends $\widetilde{ch}$ to Bob; otherwise, he aborts.
   (c) Outputs m_c and halts.
4. Upon receiving $(sid, \widetilde{ch})$ from Bob, Alice verifies that $\widetilde{ch} = ch$. If this check succeeds, Alice outputs $\perp$ and halts. Otherwise, Alice aborts.

[a] We assume an efficiently computable binary representation with $|\mathcal{R}|$ bits of elements $\mathsf{r}'_i \in \mathcal{R}$ and abuse notation by referring to that representation when writing $u_i = w'_i \oplus (w_{1-i}|\mathsf{r}'_{1-i})$.

[b] Notice that the messages in steps 2(b) and 2(c) can be sent to Bob as a single message.

Figure 3: Protocol π_{OT}

both are corrupted, $\mathcal{S}$ simply runs $\mathcal{A}$ internally. In this case, $\mathcal{A}$ will generate the messages from both corrupted parties. If neither Alice nor Bob are corrupted, $\mathcal{S}$ runs the protocol between honest Alice and Bob internally on the inputs provided by $\mathcal{Z}$ and all messages are delivered to $\mathcal{A}$. As for the correctness, notice that p_c is encrypted under public key pk_c, for which Bob knows the corresponding secret key sk_c. Hence, Bob can successfully recover p_c from ct_c and use it to obtain p'_c from $\mathcal{F}_{\mathrm{RO}}$. This enables Bob to retrieve m_c from m'_c by computing $m_c = m'_c \oplus p'_c$.

Simulator for a corrupted Bob**:** In the case where only Bob is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}_{\mathrm{OT}}$ and an internal copy $\mathcal{A}$ of the real world adversary ($\mathcal{S}$ acts as Alice in this

internal simulated execution of the protocol). Additionally, $\mathcal{S}$ also plays the role of $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ to $\mathcal{A}$. The goal of the simulator is to extract Bob's choice bit in order to request the correct message from $\mathcal{F}_{\text{OT}}$. In order to do so, $\mathcal{S}$ first executes the same steps of an honest Alice in π_{OT} with the difference that it picks uniformly random values for m'_0, m'_1 instead of computing them. Since $\mathcal{S}$ plays the role of $\mathcal{F}_{\text{RO2}}$ and $\mathcal{F}_{\text{RO3}}$ to $\mathcal{A}$, it learns Bob's choice bit when it first receives a query (sid, p_c) to $\mathcal{F}_{\text{RO2}}$ or (sid, w_c) to $\mathcal{F}_{\text{RO3}}$ from $\mathcal{A}$. Since $\mathcal{A}$ can only query the random oracle $\mathcal{F}_{\text{RO1}}$ in a polynomial number of points to obtain q, an $\mathcal{A}$ that queries $\mathcal{F}_{\text{RO2}}$ with p_{1-c} or with w_{1-c} before w_c (thus tricking $\mathcal{S}$ into extracting the wrong choice bit) can be trivially used to break Property 2.2. Knowing c, $\mathcal{S}$ obtains m_c from $\mathcal{F}_{\text{OT}}$ and answers $\mathcal{A}$'s query to $\mathcal{F}_{\text{RO2}}$ with (sid, p'_c) such that $p'_c = m'_c \oplus m_c$. $\mathcal{S}$ instructs $\mathcal{F}_{\text{OT}}$ to deliver the delayed output $(\mathsf{received}, sid)$ to Alice after it receives $\widetilde{ch} = ch$ from $\mathcal{A}$. If $\mathcal{S}$ receives an invalid $\widetilde{ch}$ it aborts. The simulator $\mathcal{S}$ for the case where only Bob is corrupted is presented in Figure 4.

Simulator $\mathcal{S}$ (Corrupted Bob)

Let λ be the length of the messages and κ be the security parameter. The simulator $\mathcal{S}$ interacts with an environment $\mathcal{Z}$, functionality $\mathcal{F}_{\text{OT}}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Bob, proceeding as follows:

1. $\mathcal{S}$ simulates the answers to the random oracle queries from $\mathcal{A}$ exactly as $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ would (and stores the lists of queries/answers), except when stated otherwise in $\mathcal{S}$'s description.
2. Upon receiving (sid, s, pk_0) from $\mathcal{A}$, $\mathcal{S}$ proceeds as follows:
   (a) Check if there is a pair (s, q) in the simulated list of $\mathcal{F}_{\text{RO1}}$. If not, $\mathcal{S}$ samples $q \xleftarrow{\$} \mathcal{PK}$ and stores (s, q) in it. $\mathcal{S}$ computes pk_1 such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$.
   (b) Sample $m'_0, m'_1 \xleftarrow{\$} \{0,1\}^\lambda$, $p_0, p_1 \xleftarrow{\$} \{0,1\}^\kappa$. If p_0 or p_1 has already been queried to $\mathcal{F}_{\text{RO2}}$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ computes $\mathsf{ct}_0 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_0, p_0)$ and $\mathsf{ct}_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, p_1)$, and sends $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ to $\mathcal{A}$.
   (c) Follow the steps of an honest Alice in π_{OT} to generate $(sid, (u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1))$ and send it to Bob, storing ch, w_0, w_1.
3. $\mathcal{S}$ sets $c = \perp$ and resumes answering queries from $\mathcal{A}$ to $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ as before, except for the following cases:
   - Upon receiving the *first*[a] query $(sid, w_{\hat{c}})$ from $\mathcal{A}$ to $\mathcal{F}_{\text{RO3}}$ where $\hat{c} \in \{0,1\}$, if $c \neq \perp$ and $c \neq \hat{c}$ (meaning that a different c was extracted from a query $(sid, p_{\tilde{c}})$ to $\mathcal{F}_{\text{RO2}}$), output fail and halt. If $c = \perp$, set $c = \hat{c}$ and send $(\mathsf{receiver}, sid, c)$ to $\mathcal{F}_{\text{OT}}$, receiving $(\mathsf{received}, sid, m_c)$ in response. $\mathcal{S}$ answers all queries from $\mathcal{A}$ to $\mathcal{F}_{\text{RO3}}$ exactly as $\mathcal{F}_{\text{RO3}}$ would.
   - Upon receiving a query $(sid, p_{\tilde{c}})$ from $\mathcal{A}$ to $\mathcal{F}_{\text{RO2}}$ where $\tilde{c} \in \{0,1\}$, if $c \neq \perp$ and $c \neq \tilde{c}$ (meaning that a different c was extracted from a query $(sid, w_{\hat{c}})$ to $\mathcal{F}_{\text{RO3}}$), output fail and halt. If $c = \perp$, set $c = \tilde{c}$, send $(\mathsf{receiver}, sid, c)$ to $\mathcal{F}_{\text{OT}}$, receiving $(\mathsf{received}, sid, m_c)$ in response. Answer the query with $(sid, m'_c \oplus m_c)$. After this point, if $\mathcal{A}$ queries (sid, p_{1-c}) to $\mathcal{F}_{\text{RO2}}$, $\mathcal{S}$ outputs fail and halts.
4. Upon receiving $(sid, \widetilde{ch})$ from $\mathcal{A}$, $\mathcal{S}$ aborts if $\widetilde{ch} \neq ch$. If $\widetilde{ch} = ch$, $\mathcal{S}$ instructs $\mathcal{F}_{\text{OT}}$ to deliver the delayed output $(\mathsf{received}, sid)$ to Alice.
5. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

[a] If a previous query $(sid, w_{1-\hat{c}})$ where $\hat{c} \in \{0,1\}$ has been received, then $\mathcal{S}$ simply answers the query as $\mathcal{F}_{\text{RO3}}$ would and skips this step.

Figure 4: Simulator $\mathcal{S}$ for the case where only Bob is corrupted.

Notice that unless the simulator $\mathcal{S}$ outputs fail, it perfectly emulates the execution of the real protocol for $\mathcal{A}$. Hence, an environment can distinguish the real world execution from the ideal

world simulation only if: (1) p_0 or p_1 is queried to $\mathcal{F}_{\text{RO2}}$ before the simulator sends the encrypted messages; (2) $\mathcal{A}$ queries both p_0 or p_1 to $\mathcal{F}_{\text{RO2}}$; or (3) $\mathcal{A}$ queries both p_c to $\mathcal{F}_{\text{RO2}}$ and w_{1-c} before w_c to $\mathcal{F}_{\text{RO3}}$. The first event can only happen with negligible probability as p_0 and p_1 are uniformly random strings of length κ and the PPT adversary $\mathcal{A}$ can only make a polynomial number of queries to the random oracle. The probabilities of the second and third events are polynomially related with the probability of breaking Property 2.2. Notice that p_c and w_c (resp. p_{1-c} and w_{1-c}) are encrypted under pk_c (resp. pk_{1-c}), so $\mathcal{A}$ must break Property 2.2 in order to obtain both p_c and p_{1-c} or p_c and w_{1-c} before w_c. Thus, a PPT environment $\mathcal{Z}$ cannot distinguish an interaction with $\mathcal{S}$ in the ideal world from an interaction with $\mathcal{A}$ in the real world except with negligible probability.

Simulator for a corrupted Alice: In the case where only Alice is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}_{\text{OT}}$ and an internal copy $\mathcal{A}$ of the real world adversary ($\mathcal{S}$ acts as Bob in this internal simulated execution of the protocol). Additionally, $\mathcal{S}$ also plays the role of $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ to $\mathcal{A}$. The goal of the simulator is to extract both messages m_0, m_1 of the receiver in order to deliver them to $\mathcal{F}_{\text{OT}}$. In order to do so, $\mathcal{S}$ has to trick $\mathcal{A}$ into accepting two public-keys $\mathsf{pk}_0, \mathsf{pk}_1$ for which $\mathcal{S}$ knows the corresponding secret-keys $\mathsf{sk}_0, \mathsf{sk}_1$. $\mathcal{S}$ generates two secret and public-key pairs $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$. Additionally, $\mathcal{S}$ generates a random seed $s \xleftarrow{\$} \{0,1\}^\kappa$ and sends (s, pk_0) to Bob. When $\mathcal{A}$ queries $\mathcal{F}_{\text{RO1}}$ with (sid, s), $\mathcal{S}$ answers with $(sid, \mathsf{pk}_0 \oplus \mathsf{pk}_1)$, thus making $\mathcal{A}$ fix pk_1 for which $\mathcal{S}$ knows the corresponding secret-key sk_1. Upon receiving $(m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ from $\mathcal{A}$, $\mathcal{S}$ uses sk_0 and sk_1 to decrypt both ct_0 and ct_1 and obtain p_0 and p_1 (respectively), thus enabling it to recover both messages of Alice. If the decryption of ct_i fails, $\mathcal{S}$ samples a random $m_i \xleftarrow{\$} \{0,1\}^\lambda$. Moreover, $\mathcal{S}$ has to check that $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$ was correctly constructed by executing the checks of an honest Bob in Protocol π_{OT}. Notice that these checks will fail regardless of the choice bit of Bob in case $\mathcal{A}$ provides an invalid message, allowing $\mathcal{S}$ to run these checks with either sk_0 or sk_1. The simulator $\mathcal{S}$ for the case where only Alice is corrupted is presented in Figure 5.

The only points where the simulation differs from the real world protocol execution are in the sampling of two public-keys $\mathsf{pk}_0, \mathsf{pk}_1$ for which $\mathcal{S}$ knows the corresponding secret-keys, in the answer $(sid, \mathsf{pk}_0 \oplus \mathsf{pk}_1)$ to the query (sid, s) from $\mathcal{A}$ to (the simulated) $\mathcal{F}_{\text{RO1}}$ and in the checks performed by the simulator on $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$. Notice that due to Property 2.3, a public-key outputted by KG is indistinguishable from a key sampled uniformly at random from $\mathcal{PK}$. Hence, the choice of public-keys in the simulation is indistinguishable from that of the real world protocol execution. The only way the simulation can still fail is if $\mathcal{A}$ queries $\mathcal{F}_{\text{RO1}}$ with (sid, s) before $\mathsf{pk}_0, \mathsf{pk}_1$ are chosen by $\mathcal{S}$, which happens with negligible probability. The difference in the checking of $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$ is that the simulator always uses sk_0 instead of using sk_c. Notice that due to Property 2.4, the adversary cannot provide a ciphertext ct'_1 and u_1 such that these checks would yield a different result if executed using sk_1. Thus, a PPT environment $\mathcal{Z}$ cannot distinguish an interaction with $\mathcal{S}$ in the ideal world from an interaction with $\mathcal{A}$ in the real world except with negligible probability. ▌

3.2 Obtaining 1-out-of-k OT

Our OT protocol can be easily extend to obtain 1-out-of-k OT, i.e., Alice has k input strings and Bob can learn one of the strings that he chooses. The idea is that there are k public-keys $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{k-1})$ that are used to encrypted the strings using the same technique as before. In the modified protocol, Bob generates a pair of keys $(\mathsf{pk}_c, \mathsf{sk}_c) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ for his choice

Simulator $\mathcal{S}$ (Corrupted Alice)

Let λ be the length of messages and κ be a security parameter. Simulator $\mathcal{S}$ interacts with an environment $\mathcal{Z}$, functionality $\mathcal{F}_{\text{OT}}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Alice, proceeding as follows:

1. $\mathcal{S}$ simulates the answers to the random oracle queries from $\mathcal{A}$ exactly as $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ would (and stores the lists of queries/answers), except when stated otherwise in $\mathcal{S}$'s description.
2. $\mathcal{S}$ generates two secret and public-key pairs $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$. $\mathcal{S}$ samples $s \xleftarrow{\$} \{0,1\}^\kappa$ and aborts if s has been already queried to $\mathcal{F}_{\text{RO1}}$. $\mathcal{S}$ sends (s, pk_0) to $\mathcal{A}$.
3. When $\mathcal{A}$ queries $\mathcal{F}_{\text{RO1}}$ with (sid, s), $\mathcal{S}$ answers with $(sid, \mathsf{pk}_0 \oplus \mathsf{pk}_1)$.
4. Upon receiving $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ and $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$, $\mathcal{S}$ proceeds as follows:
   (a) decrypts ct_0 and ct_1, getting $p_0 \leftarrow \mathsf{Dec}(\mathsf{sk}_0, \mathsf{ct}_0)$ and $p_1 \leftarrow \mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_1)$. If the decryption of ct_i failed, $\mathcal{S}$ samples $m_i \xleftarrow{\$} \{0,1\}^\lambda$. Otherwise, $\mathcal{S}$ recovers p'_i from the list (p_i, p'_i) stored for $\mathcal{F}_{\text{RO2}}$ (or samples $p'_i \xleftarrow{\$} \{0,1\}^\lambda$ and stores the new pair if such a pair does not exist yet) and computes both messages $m_0 = m'_0 \oplus p'_0$ and $m_1 = m'_0 \oplus p'_0$.
   (b) uses sk_0 to execute the steps of an honest Bob in Protocol π_{OT} to verify that $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$ was correctly constructed by $\mathcal{A}$ and compute $\widetilde{ch}$. In case the checks succeeds[a], $\mathcal{S}$ sends $(sid, \widetilde{ch})$ to $\mathcal{A}$. Otherwise, it aborts.
   (c) sends $(\mathsf{sender}, sid, m_0, m_1)$ to $\mathcal{F}_{\text{OT}}$.
5. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

[a]Notice these checks will fail regardless of the choice bit c if this message is invalid.

Figure 5: Simulator $\mathcal{S}$ for the case where only Alice is corrupted.

value $c \in \{0, \ldots, k-1\}$. He also samples a random string $s \xleftarrow{\$} \{0,1\}^\kappa$ and sends the queries $(sid, s\|1), \ldots, (sid, s\|k-1)$ to $\mathcal{F}_{\text{RO1}}$, obtaining $(sid, q_1), \ldots, (sid, q_{k-1})$ as answers. Bob then computes all pk_i for $i \in \{0, \ldots, k-1\} \setminus c$ in such way that $\mathsf{pk}_0 \star \mathsf{pk}_j = q_j$ for $j \in \{1, \ldots, k-1\}$. He sends (s, pk_0) to Alice. She uses the answers to the same random oracle queries as well as pk_0 to reconstruct $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{k-1})$.

4 Adaptive Security

In this section we show that Protocol π_{OT} is secure against adaptive adversaries. In the case of adaptive corruptions, the simulator has to handle adversaries that corrupt parties after the protocol execution has started, potentially after the execution is finished. When a party is corrupted in the ideal world, the simulator learns its inputs (and possibly outputs) and needs to hand it to its internal copy of the adversary along with the internal state of the dummy party (run internally by the simulator) corresponding to the corrupted party. This internal state must be consistent with both the inputs learned upon corruption and the messages already sent between dummy parties in the simulation. Usually it is hard to construct a simulator capable of doing so because it must first simulate a protocol execution with its internal copy of the adversary without knowing the inputs of uncorrupted parties and later, if a corruption happens, it must generate an internal state for the corrupted dummy party that is consistent with the newly learned inputs and the protocol messages that have already been generated. Intuitively, this requirement means that the simulator must simulate "non-committing" messages for honest dummy parties such that, upon corruption, it can generate randomness that would lead an honest party executing the protocol to generate the messages sent up to that point had it been given the inputs obtained

form the ideal world party.

In order to prove that Protocol π_{OT} is indeed secure against adaptive adversaries, we will construct a simulator that generates messages in its internal execution with a copy of the adversary such that it can later come up with randomness that would result in these messages being generated given any input to any of the parties. The general structure of this simulator is very similar to the simulator for the static case, using the same techniques for extracting inputs. The main modification in the simulator lies in the case where both the Alice and Bob are honest. In the proof of security against adaptive adversaries, the simulator no longer handles this case by simulating an interaction between two internal dummy parties running the protocol on random inputs. Instead, the simulator generates the first message (from Bob to Alice) by acting as in the static case of a corrupted sender and the second message (from Alice to Bob) by acting as in the static case of a corrupted receiver. From the proof of static security, this clearly generates a view that is indistinguishable from a real execution of the protocol. However, now the first message (s, pk_0) is such that both pk_0 and pk_1 (as defined by the protocol) are valid public-keys. Similarly, the second message $(m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ contains random m'_0, m'_1 and $\mathsf{ct}_0, \mathsf{ct}_1$ containing p_0, p_1 that haven't yet been queried to $\mathcal{F}_{\text{RO2}}$. Notice that the states and messages of steps 2(c), 3(b) and 4 of Protocol π_{OT} does not depend on the inputs of Alice or Bob, so that it suffices for $\mathcal{S}$ to simulate them by acting exactly as honest parties would in π_{OT}.

Having these messages generated by simulating an execution between honest Alice and Bob will allow the simulator to "explain" the randomness used in an execution in case of an adaptive corruption. In the case of an adaptive corruption of Bob, the simulator can hand the state of Bob upon corruption to its internally executed adversary by simply giving the adversary c,$(\mathsf{pk}_c, \mathsf{sk}_c)$,$s$,$(sid, q)$, pk_{1-c} and the random coins of $\mathsf{KG}(1^\kappa)$ when generating $(\mathsf{pk}_c, \mathsf{sk}_c)$, given c revealed when the the ideal world Bob is corrupted. Since both pk_0 and pk_1 are valid and a valid public-key is indistinguishable from an invalid one, the state is consistent with c. In the case of an adaptive corruption of Alice, the simulator obtains (m_0, m_1) by corrupting the ideal world Alice, sets the answers (sid, p'_0) and (sid, p'_1) to queries (sid, p_0) and (sid, p_1) to the simulated $\mathcal{F}_{\text{RO2}}$ such that $m'_0 = p'_0 \oplus m_0, m'_1 = p'_1 \oplus m_1$, and finally hands Alice's state to the adversary as m_0, m_1, pk_0, pk_1 (computed as in the protocol), p_0, p_1, p'_0, p'_1 and message $(m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$.

Theorem 4.1 *Let* PKE *be a public-key encryption scheme that satisfies Properties 2.2, 2.3 and 2.4. When instantiated with* PKE*, Protocol π_{OT} UC-realizes the functionality $\mathcal{F}_{\text{OT}}$ against adaptive adversaries in the $\mathcal{F}_{\text{RO}}$-hybrid model.*

Proof: As in the case of an static adversary, we must construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between interactions with an adversary $\mathcal{A}$ in the real world and with $\mathcal{S}$ in the ideal world. For the sake of clarity, we will describe the simulator $\mathcal{S}$ separately for the different corruption scenarios. In all cases, $\mathcal{S}$ 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 $\mathcal{Z}$.

First we handle the four cases corresponding to the corruption statuses (*i.e.* corrupted or honest) of Alice and Bob before the execution starts:

Case 1: Both Alice and Bob are honest: This case is the main point where the simulator for an adaptive adversary changes in relation to the static adversary case. In this case, the simulator will simulate an execution between honest Alice and Bob using random inputs. The easiest way to perform this simulation is to run the protocol exactly as honest parties would when given

random inputs, which is the done in the proof of Theorem 3.1. However, for handling adaptive corruptions, the simulator will generate "non-committing" messages for honest dummy parties in such a way that the simulated execution is indistinguishable from a real execution and that the simulator can later generate consistent randomness to claim that honest parties had been executing the protocol with an arbitrary input. The simulator for this case is described in Figure 6.

Simulator $\mathcal{S}$ (Honest Alice and Bob)

Let λ be the length of the messages and κ be the security parameter. The simulator $\mathcal{S}$ interacts with an environment $\mathcal{Z}$, functionality $\mathcal{F}_{\text{OT}}$ and an internal copy $\mathcal{A}$ of the adversary that does not corrupt Alice or Bob, only observing their interaction and querying $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$. $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ simulates the answers to the random oracle queries from $\mathcal{A}$ exactly as $\mathcal{F}_{\text{RO1}}$, $\mathcal{F}_{\text{RO2}}$, $\mathcal{F}_{\text{RO3}}$ and $\mathcal{F}_{\text{RO4}}$ would (and stores the lists of queries/answers), except when stated otherwise in $\mathcal{S}$'s description.
2. Simulating the first round:
   (a) $\mathcal{S}$ generates two secret and public key pairs $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ using randomness $\mathsf{KG}.\mathsf{r}_0$ and $\mathsf{KG}.\mathsf{r}_1$, respectively. $\mathcal{S}$ samples $s \xleftarrow{\$} \{0,1\}^\kappa$, outputting fail and halting if s has been already queried to $\mathcal{F}_{\text{RO1}}$. $\mathcal{S}$ sends (s, pk_0) to the dummy Alice.
   (b) When $\mathcal{A}$ queries $\mathcal{F}_{\text{RO1}}$ with (sid, s), $\mathcal{S}$ answers with $(sid, \mathsf{pk}_0 \oplus \mathsf{pk}_1)$.
3. Simulating the second round:
   (a) $\mathcal{S}$ samples $m'_0, m'_1 \xleftarrow{\$} \{0,1\}^\lambda$, $p_0, p_1 \xleftarrow{\$} \{0,1\}^\kappa$ and $\mathsf{r}_0, \mathsf{r}_1 \xleftarrow{\$} \mathcal{R}$. If p_0 or p_1 has already been queried to $\mathcal{F}_{\text{RO2}}$, $\mathcal{S}$ outputs fail and halts. Otherwise, $\mathcal{S}$ computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}_0, p_0, \mathsf{r}_0)$ and $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}_1, p_1, \mathsf{r}_1)$, and sends $(m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ to dummy Bob.
   (b) $\mathcal{S}$ follows the steps of an honest Alice in π_{OT} to generate $(sid, (u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1))$ and send it to dummy Bob, storing ch, w_0, w_1.
   (c) $\mathcal{S}$ resumes answering the random oracle queries from $\mathcal{A}$ as above, except for the queries p_0 and p_1 to $\mathcal{F}_{\text{RO2}}$. Upon receiving such query p_c, $\mathcal{S}$ outputs fail and aborts.
4. Simulating the third round:
   (a) $\mathcal{S}$ sends $(sid, \widetilde{ch})$ to dummy Alice, where $\widetilde{ch} = ch$. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

Figure 6: Simulator $\mathcal{S}$ for the case where both Alice and Bob are honest.

Case 2: Alice is corrupted and Bob is honest: This case is equivalent to an execution with a static adversary that corrupts only Alice. Hence, the simulator proceeds as in the proof of Theorem 3.1.

Case 3: Alice is honest and Bob is corrupted: This case is equivalent to an execution with a static adversary that corrupts only Bob. Hence, the simulator proceeds as in the proof of Theorem 3.1.

Case 4: Both Alice and Bob are corrupted: This case is equivalent to an execution with a static adversary that corrupts both Alice and Bob. Hence, the simulator proceeds as in the proof of Theorem 3.1.

Now we analyze the adaptive corruption cases. Notice that the simulator proceeds to simulate as in the static case after handing the internal state of the corrupted dummy party to the ad-

versary. This is the case because the messages generated by the simulator for honest dummy parties already allow the simulator to continue simulating using the same instructions. In the case of a corrupted Alice, the message (s, pk_0) generated for the dummy receiver is already such that both pk_0 and pk_1 (if computed as in the protocol) are valid public-keys for which the simulator knows the corresponding secret-keys, allowing it to extract the adversary's messages. In the case of a corrupted Bob, the message $(m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ generated for the dummy sender is already such that p_0 or p_1 have not been queried to $\mathcal{F}_{\text{RO2}}$, allowing the simulator to set the answer to these queries in such a way that $m_c = m'_c \oplus p'_c$ for any arbitrary $m_c \in \{0,1\}^\lambda$ and any $c \in \{0,1\}$. Hence, we focus describing how the simulator generates the internal states of dummy Alice and dummy Bob in case the adversary chooses to corrupt them during the simulation. We will handle the cases of adaptive corruptions of Alice and Bob separately.

Adaptive corruption of Alice: When Alice is corrupted in the ideal world, $\mathcal{S}$ obtains its input (m_0, m_1). In the simulated execution with its internal copy of the adversary $\mathcal{A}$, $\mathcal{S}$ must return both the input and a consistent internal state for the dummy Alice. We further divide this scenario in two cases regarding the point of the execution when Alice becomes corrupted:

- Before the second round: In this case, dummy Alice's internal state consists solely of the inputs (m_0, m_1). $\mathcal{S}$ returns (m_0, m_1) to $\mathcal{A}$.
- After the second round: In this case, $\mathcal{S}$ must return to $\mathcal{A}$ both the input (m_0, m_1) and the internal state of dummy Alice in a way that it is consistent with both the input and the messages $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ and $(sid, (u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1))$ sent by dummy Alice. In both the cases of an honest Bob and of a dishonest Bob, these messages are computed by $\mathcal{S}$ in the same way and $\mathcal{S}$ knows p_0, p_1. Hence, $\mathcal{S}$ computes $p'_0 = m'_0 \oplus m_0$, $p'_1 = m'_1 \oplus m_1$. $\mathcal{S}$ returns $(m_0, m_1, \mathsf{pk}_1, p_0, p_1, p'_0, p'_1, \mathsf{r}_0, \mathsf{r}_1, w_0, w_1, w'_0, w'_1, \mathsf{r}'_0, \mathsf{r}'_1, ch)$ to $\mathcal{A}$ as the dummy Alice's state. Furthermore, $\mathcal{S}$ answers with $(sid, m'_i \oplus m_i)$ if $\mathcal{A}$ queries $\mathcal{F}_{\text{RO2}}$ with p_i, for $i \in \{0,1\}$, and answers with (sid, ch) if $\mathcal{A}$ queries $\mathcal{F}_{\text{RO4}}$ with $(sid, w_0|w_1|\mathsf{r}'_0|\mathsf{r}'_1)$. Notice that the values $(w_0, w_1, w'_0, w'_1, \mathsf{r}'_0, \mathsf{r}'_1, ch)$ related to steps 2(c), 3(b) and 4 of π_{OT} do not depend on the on the inputs of Alice.

If Bob is honest, $\mathcal{S}$ continues the simulation from the point where the corruption happened as in Case 2 (where only Alice is corrupted). Otherwise, it continues as in Case 4 (where both Alice and Bob are corrupted).

Adaptive corruption of Bob: When Bob is corrupted in the ideal world, $\mathcal{S}$ obtains its input c. In the simulated execution with its internal copy of the adversary $\mathcal{A}$, $\mathcal{S}$ must return both the input and a consistent internal state for the dummy Bob. We further divide this scenario in two cases regarding the point of the execution when Bob becomes corrupted:

- Before the first round: In this case, dummy Bob's internal state consists solely of the input c. $\mathcal{S}$ returns c to $\mathcal{A}$.
- Between the first and second rounds: In this case, $\mathcal{S}$ must return to $\mathcal{A}$ both the input c and the internal state of dummy Bob in such a way that it is consistent with both the input and the first message (s, pk_0) sent by dummy Bob. In both the cases of an honest Alice and of a dishonest Alice, the first message is computed by $\mathcal{S}$ in the same way and

he knows both key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$, as well as the randomness $\mathsf{KG.r}_0$ and $\mathsf{KG.r}_1$ used to generate each of them. Hence, $\mathcal{S}$ computes $q = \mathsf{pk}_0 \oplus \mathsf{pk}_1$ and returns $(c, \mathsf{sk}_c, \mathsf{pk}_c, \mathsf{KG.r}_0, s, q, \mathsf{pk}_{1-c})$ to $\mathcal{A}$ as the internal state of dummy Bob, where r_c is the randomness used by $(\mathsf{pk}_c, \mathsf{sk}_c) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$. Notice that in the case where Alice and Bob are honest as well as in the case where only Alice is corrupted, $\mathcal{S}$ would answer a query (sid, s) to $\mathcal{F}_{\text{RO1}}$ with (sid, q), where $q = \mathsf{pk}_0 \oplus \mathsf{pk}_1$. Hence, when $\mathcal{A}$ queries $\mathcal{F}_{\text{RO1}}$ with (sid, s), $\mathcal{S}$ consistently answers with (sid, q).

- After the second round: In this case, $\mathcal{S}$ also obtains Bob's output m_c in the ideal world. Moreover, $\mathcal{S}$ has to $\mathcal{S}$ has to provide an internal state that is compatible with messages $(sid, m'_0, m'_1, \mathsf{ct}_0, \mathsf{ct}_1)$ and $(sid, u_0, u_1, \mathsf{ct}'_0, \mathsf{ct}'_1)$ received in the second round, apart from the input c and the first message (s, pk_0) sent by dummy Bob. Hence, $\mathcal{S}$ must return to $\mathcal{A}$ the input c, randomness that is consistent with the first message and, additionally, randomness that is consistent with the second message. The message in the first round does not depend on the output, so $\mathcal{S}$ computes $(\mathsf{sk}_c, \mathsf{pk}_c, \mathsf{KG.r}_0, s, q, \mathsf{pk}_{1-c})$ and handles queries to $\mathcal{F}_{\text{RO1}}$ as in the previous case. If Alice is corrupted, $\mathcal{S}$ retrieves (p_c, p'_c) from the internal list of the simulated $\mathcal{F}_{\text{RO2}}$ (since either $\mathcal{A}$ or $\mathcal{S}$ made this query). Otherwise, if Alice is honest, $\mathcal{S}$ sets $p'_c = m'_c \oplus m_c$. Finally, $\mathcal{S}$ returns $(c, \mathsf{sk}_c, \mathsf{pk}_c, \mathsf{r}_c, s, q, \mathsf{pk}_{1-c}, p_c, p'_c, w_0, w'_0, w_1, w'_1, \mathsf{r}'_0, \mathsf{r}'_1)$ to $\mathcal{A}$ as dummy Bob's internal state. Furthermore, upon receiving a query p_c from $\mathcal{A}$ to $\mathcal{F}_{\text{RO2}}$, $\mathcal{S}$ answers with the value $m'_c \oplus m_c$. Notice that the values $(w_0, w'_0, w_1, w'_1, \mathsf{r}'_0, \mathsf{r}'_1)$ related to steps 2(c), 3(b) and 4 of π_{OT} do not depend on the on the inputs of Bob, as it would obtain the same values regardless of c as argued in the proof of the static case.

If Alice is honest, $\mathcal{S}$ continues the simulation from the point where the corruption happened as in Case 3 (where only Bob is corrupted). Otherwise, it continues as in Case 4 (where both Alice and Bob are corrupted). ▮

5 Instantiations of the OT Framework

5.1 Instantiation Based on the McEliece Cryptosystem

Let n, k and t be functions of the security parameter κ (for easy of readability this will not be explicit in the notation). Consider a linear-error correcting code C with length n and dimension k, which consists of a k-dimensional subspace of $\mathbb{F}_2^n$, and let $G \in \mathbb{F}_2^{k \times n}$ denote the generator matrix of C. For the parameters n, k and t, the assumption on the hardness of the bounded decoding problem can be stated as follows.

Assumption 5.1 For parameters n, k and t that are functions of the security parameter κ, sample the generator matrix $G \xleftarrow{\$} \mathbb{F}_2^{k \times n}$, the message $x \xleftarrow{\$} \mathbb{F}_2^k$ and a uniformly random error $e \in \mathbb{F}_2^n$ such that $\mathsf{HW}(e) = t$. Then for $y = xG \oplus e$ and for every PPT decoder $\mathcal{A}$

$$\Pr[\mathcal{A}(G, y) = x] \in \mathsf{negl}(\kappa).$$

We next prove some useful facts that will be used in our constructions, these observations follow the lines of Mathew et al. [54]. Lets call a fixed generator matrix G good if there exists a PPT decoder that can recover the message x with non-negligible probability when encoded with this specific G. Clearly if the assumption on the hardness of the bounded decoding problem holds, then the subset of good generator matrices constitutes only a negligible fraction of all generator matrices.

Observation 5.2 [[54]] The fraction of matrices $Q \in \mathbb{F}_2^{k\times n}$ that can be expressed as $Q = G_1 \oplus G_2$ for good generator matrices G_1 and G_2 is negligible.

Proof: This follows from a simple counting argument: if there are g good generator matrices, there can be at most $\binom{g}{2} = O(g^2)$ matrices Q that can be expressed as the exclusive-or of two good generator matrices. ▌

One possible instantiation of our OT protocol uses the McEliece cryptosystem, which has $\mathcal{PK} = \mathbb{F}_2^{k\times n}$, $\mathcal{M} = \mathbb{F}_2^k$, $\mathcal{C} = \mathbb{F}_2^n$, and works as follows:

- Key generation: Generate a generator matrix $G \in \mathbb{F}_2^{k\times n}$ of a Goppa code together with the efficient error-correction algorithm $\mathsf{Correct}$ that can correct up to t errors. Generate a uniformly random non-singular matrix $S \in \mathbb{F}_2^{k\times k}$ and a uniformly random permutation matrix $T \in \mathbb{F}_2^{n\times n}$. Set $\mathsf{pk} = SGT$ and $\mathsf{sk} = (S, G, T)$.
- Encryption: Given as input the public-key pk and the message $\mathsf{m} \in \mathbb{F}_2^k$, sample uniformly at random an error vector $e \in \mathbb{F}_2^n$ such that $\mathsf{HW}(e) = t$ and output the ciphertext $\mathsf{ct} \leftarrow \mathsf{m} \cdot \mathsf{pk} \oplus e$.
- Decryption: Given the ciphertext ct and the secret-key sk as input, compute $\mathsf{ct} \cdot T^{-1} = (\mathsf{m}S)G \oplus eT^{-1}$. Then compute $\mathsf{m}S \leftarrow \mathsf{Correct}(cT^{-1})$ and output $\mathsf{m} = (\mathsf{m}S)S^{-1}$.

The security of the McEliece cryptosystem relies on Assumption 5.1. Therefore, when considering the operation $\oplus$ on $\mathcal{PK}$, Observation 5.2 implies that Property 2.2 holds straightforwardly if Assumption 5.1 is true. The security of the McEliece encryption scheme is also based on the following assumption regarding the pseudorandomness of the public-keys, which is equivalent to Property 2.3 for this cryptosystem.

Assumption 5.3 Let $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\kappa)$ be the key generation algorithm of the McEliece cryptosystem for $\mathsf{pk} \in \mathbb{F}_2^{k\times n}$. Let $R \xleftarrow{\$} \mathbb{F}_2^{k\times n}$. For every PPT distinguisher $\mathcal{A}$ that outputs a decision bit it holds that

$$|\Pr[\mathcal{A}(\mathsf{pk}) = 1] - \Pr[\mathcal{A}(R) = 1]| \in \mathsf{negl}(\kappa).$$

Property 2.4 follows from the fact that the amount of noise inserted in the codeword is bellow the error capability of the underlying error correcting code. Therefore, only one codeword will be within the decoding sphere associated with a single ciphertext.

5.2 QC-MDPC based Instantiation

The QC-MDPC cryptosystem [64] is a variant of the McEliece cryptosystem that has much shorter keys. Currently, it is the most efficient code-based public-key cryptosystem. We briefly describe it here. In the following $\mathrm{wt}(x)$ denotes the Hamming weight of a vector x.

- Key generation: Let $\mathrm{circ}(v)$ denote the circulant matrix whose first row is $v \in \mathbb{F}_2^r$, a binary array of length r. A QC-MDPC secret-key is a sparse parity-check matrix of form $\hat{H} = [\mathrm{circ}(f) \mid \mathrm{circ}(g)]$ where $\gcd(g, x^r - 1) = 1$ and $\mathrm{wt}(f) + \mathrm{wt}(g) = t$ (for some suitable choice of r and t), and the corresponding public-key is the systematic parity-check matrix $H = [\mathrm{circ}(h) \mid I]$ where $h = f \cdot g^{-1} \mod x^r - 1$, or equivalently the systematic generator $G = \left[I \mid \mathrm{circ}(h)^T\right]$, both represented by $h \in \mathbb{F}_2^r$. .

- Encryption: We encrypt a message by encoding it as a vector $e = (e_0, e_1) \in \mathbb{F}_2^{2r}$ of weight $\mathrm{wt}(e) = t$, choosing a uniformly random $m \xleftarrow{\$} \mathbb{F}_2^r$ and computing the ciphertext as $c \leftarrow mG + e \in \mathbb{F}_2^{2r}$.

- Decryption: Let $\Psi_{\hat{H}}$ denote a t-error correcting decoding algorithm that has knowledge of the secret-key $\hat{H}$. Extract the error vector e by decoding $c \leftarrow \Psi_{\hat{H}}(c)$.

Assumption 5.4 The *decisional QC-MDPC assumption* states that distinguishing h from a uniformly random vector $u \in \mathbb{F}_2^r$ is unfeasible.

The OW-CPA security of this cryptosystem follows from the pseudorandomess of the public-key (the decisional QC-MDPC assumption) and the hardness of the bounded decoding problem (Assumption 5.1), exactly as in the original McEliece.

The QC-MDPC cryptosystem as described here does satisfy the requirements to be used within our general construction. The analysis is exactly the same as in the original McEliece PKC in Section 5.1 and boils down to two arguments:

- The fraction of vectors $q \in \mathbb{F}_2^r$ that can be expressed as $q = h_1 \oplus h_2$ for good vectors h_1 and h_2 is negligible. Here, a good vector $h_i, i \in \{0, 1\}$ is one where the corresponding parity-check matrix $H_i = [\mathrm{circ}(h) \mid I]$ has a trapdoor. This fact and the hardness of the bounded decoding problem (Assumption 5.1) imply Property 2.2.

- The pseudorandomness of the public-key implies Property 2.3.

As in the original McEliece cryptosystem, property 2.4 also follows from the fact that the amount of noise inserted in the codeword is bellow the error capability of the underlying error correcting code. Therefore, only one codeword will be within the decoding sphere associated with a single ciphertext.

5.3 LPN-based Instantiation

We also present a solution based on the Learning Parity with Noise (LPN) problem. This problem essentially states that given a system of binary linear equations in which the outputs are disturbed by some noise, it is difficult to determine the solution. In this work we use the low noise variant of LPN as first studied by Alekhnovich in [2]. The following equivalent statement of the assumption is from Döttling et al. [33].

Assumption 5.5 Let the problem parameter be $n \in \mathbb{N}$, which is a function of the security parameter κ. Let $m = O(n)$, $\epsilon > 0$ and $\rho = \rho(n) = O(n^{-1/2-\epsilon})$. Sample $B \xleftarrow{\$} \mathbb{F}_2^{m \times n}$, $x \xleftarrow{\$} \mathbb{F}_2^n$ and $e \xleftarrow{\$} \chi_\rho^m$. The problem is, given B and $y \in \mathbb{F}_2^m$, to decide whether $y = Bx + e$ or $y \xleftarrow{\$} \mathbb{F}_2^m$. The assumption states that for every PPT distinguisher $\mathcal{A}$ that outputs a decision bit it holds that

$$|\Pr[\mathcal{A}(B, y = Bx + e) = 1] - \Pr[\mathcal{A}(B, y \xleftarrow{\$} \mathbb{F}_2^m) = 1]| \in \mathsf{negl}(\kappa).$$

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

Our OT protocol can be instantiated based on the IND-CPA secure cryptosystem of Döttling et al. [33] that is based on the low noise variant of LPN. We should emphasize that we just need

the simplest version of their cryptosystem, which is described below. The IND-CPA security of their cryptosystem is based on Assumption 5.5.

Consider the security parameter κ, and let $n, \ell_1, \ell_2 \in O(\kappa^{2/(1-2\epsilon)})$ and $\rho \in O(\kappa^{-(1+2\epsilon)/(1-2\epsilon)})$ so that Assumption 5.5 is believed to hold. Let $G \in \mathbb{F}_2^{\ell_2 \times n}$ be the generator-matrix of a binary linear error-correcting code C and Decode_C an efficient decoding procedure for C that corrects up to $\alpha\ell_2$ errors for constant α.

- Key Generation: Let $A \xleftarrow{\$} \mathbb{F}_2^{\ell_1 \times n}$, $T \xleftarrow{\$} \chi_\rho^{\ell_2 \times \ell_1}$ and $X \xleftarrow{\$} \chi_\rho^{\ell_2 \times n}$. Set $B = TA + X$, $\mathsf{pk} = (A, B)$ and $\mathsf{sk} = T$. Output $(\mathsf{pk}, \mathsf{sk})$.
- Encryption: Given a message $\mathsf{m} \in \mathbb{F}_2^n$ and the public-key $\mathsf{pk} = (A, B)$ as input, sample $s \xleftarrow{\$} \chi_\rho^n$, $e_1 \xleftarrow{\$} \chi_\rho^{\ell_1}$ and $e_2 \xleftarrow{\$} \chi_\rho^{\ell_2}$. 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: Given a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and a secret-key $\mathsf{sk} = T$ as input, compute $y \leftarrow \mathsf{ct}_2 - T\mathsf{ct}_1$ and $\mathsf{m} \leftarrow \mathsf{Decode}_C(y)$. Output m.

If Assumption 5.5 holds, then Property 2.3 trivially holds for this cryptosystem as the public-keys are pseudorandom [33]. Based on Assumption 5.5, Döttling et al. [33] also proved that this cryptosystem is IND-CPA secure [33] (which is a stronger notion than OW-CPA). By letting $\mathcal{PK} = \mathbb{F}_2^{\ell_1 \times n} \times \mathbb{F}_2^{\ell_2 \times n}$, considering the operation $\oplus$ on $\mathcal{PK}$ and using a counting argument about the good public-keys as in Section 5.1, we obtain that Property 2.2 trivially holds for this cryptosystem if Assumption 5.5 holds.

Property 2.4 holds for the same reason as stated in the case of the McEliece cryptosystem: the noise $Xs + e_2$ added to the codeword is within the error correction capabilities of the code C.

5.4 Instantiation based on the CDH assumption

We will instantiate our scheme by showing that the ElGamal cryptosystem is OW-CPA secure and has Properties 2.2 and 2.3 under the Computational Diffie-Hellman (CDH) assumption. First we will recall the CDH assumption:

Assumption 5.6 The Computational Diffie-Hellman assumption requires that for every PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\mathcal{A}(\mathbb{G}, w, g, g^a, g^b) = g^{ab}] \in \mathsf{negl}(\kappa).$$

where the probability is taken over the experiment of generating a group $\mathbb{G}$ of order w with a generator g on input 1^κ and choosing $a, b \xleftarrow{\$} \mathbb{Z}_q$.

The classical ElGamal cryptosystem [44] is parametrized by a group $(\mathbb{G}, g, w)$ of order w with generator g where the CDH assumption holds. We assume that $(\mathbb{G}, g, w)$ is known by all parties. The cryptosystem consists of a triple of algorithms $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ that proceed as follows:

- KG samples $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_w$, computes $\mathsf{pk} = g^{\mathsf{sk}}$ and outputs a secret and public-key pair $(\mathsf{pk}, \mathsf{sk})$.
- Enc takes as input a public-key pk and a message $\mathsf{m} \in \mathbb{G}$, samples $\mathsf{r} \xleftarrow{\$} \mathbb{Z}_p$ computes $c_1 = g^{\mathsf{r}}$, $c_2 = m \cdot \mathsf{pk}^{\mathsf{r}}$ and outputs a ciphertext $\mathsf{ct} = (c_1, c_2)$.

- Dec takes as input a secret-key sk, a ciphertext ct and outputs a message $\mathsf{m} = c_2/c_1^{\mathsf{sk}}$.

The ElGamal cryptosystem described above is well-known to be OW-CPA secure [44], leaving us to prove that it has Properties 2.2 and 2.3. Property 2.3 follows trivially from the fact that pk is chosen uniformly over all elements of $\mathbb{G}$.

Observation 5.7 The ElGamal cryptosystem described above satisfies Property 2.2 under the CDH assumption.

Proof: First we observe that $\mathcal{PK}$ is $\mathbb{G}$, which is a group. Assume by contradiction that an adversary $\mathcal{A}$ succeeds in the experiment of Property 2.2. Under the CDH assumption, $\mathcal{A}$ must know both sk_1 and sk_2 corresponding to pk_1 and pk_2. However, we know that $\mathsf{pk}_1 \cdot \mathsf{pk}_2 = q$ for a uniformly random $q \xleftarrow{\$} \mathbb{G}$ (using multiplicative notation for $\mathbb{G}$). If $\mathcal{A}$ freely generated pk_1 and pk_2 such that $\mathsf{pk}_1 \cdot \mathsf{pk}_2 = q$ and knows secret-keys sk_1 and sk_2, then it knows the discrete logarithm of q, since it is equal to $\mathsf{sk}_1 + \mathsf{sk}_2$. The CDH assumption implies that computing discrete logarithms is hard, hence we have a contradiction and the observation holds. ▌

Property 2.4 holds for ElGamal cryptosystem as the ciphertext is a one-to-one function of the plaintext and randomness.

5.4.1 An optimized CDH based instantiation

If we instantiate Protocol π_{OT} with the ElGamal cryptosystem described above in a black-box way, the sender Alice has to compute and send both ciphertexts $\mathsf{ct}_0 = (g^{\mathsf{r}_0}, p_0 \cdot \mathsf{pk}_0^{\mathsf{r}_0})$ and $\mathsf{ct}_1 = (g^{\mathsf{r}_1}, p_1 \cdot \mathsf{pk}_1^{\mathsf{r}_1})$, which amounts to 4 exponentiations and 4 group elements. The same costs are present in computing and sending ciphertexts $\mathsf{ct}'_0, \mathsf{ct}'_1$ for plaintext messages w_0, w_1. However, notice that in both cases pairs of plaintext messages m_0 and m_1 are being encrypted under two different public-keys pk_0 and pk_1, respectively. Hence, we can invoke a result by Bellare *et al.* [4] showing that each pair of ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ (resp. $\mathsf{ct}'_0, \mathsf{ct}'_1$) can be computed with the same randomness r (resp. r'). This simple observation can increase both computational and communication efficiency, since terms of the ciphertexts that depend only on the randomness only need to be computed and sent once.

Basically, the result of Bellare *et al.* [4] shows that randomness can be reused while maintaining the same underlying security guarantees when encrypting multiple messages under multiple different public-keys with "reproducible" cryptosystems, of which the ElGamal cryptosystem is an example as proven in [4]. Bellare *et al.* provide a modular generic reduction showing that it can be used to prove that reproducible cryptosystems remain IND-CPA or IND-CCA secure under randomness reuse. Since the ElGamal cryposystem is IND-CPA secure under the DDH assumption, we can directly optimize the ElGamal based instantiation above under the DDH assumption. However, we remark that the reduction of Bellare *et al.* is generic (depending only on the cryptosystem being reproducible) and straightforward to adapt to OW-CPA security, which is naturally implied by IND-CPA security. Hence, applying the techniques of Bellare *et al.* we can show that the ElGamal cryptosystem remains OW-CPA secure with randomness reuse under the CDH assumption when multiple different messages are encrypted under multiple different keys because it is a reproducible cryptosystem (as already proven in [4]). Due to space restrictions and the lack of novelty, we leave a full discussion of the application of the result of Bellare *et al.* [4] to OW-CPA secure cryptosystems to the full version of this paper.

Reusing the same randomness for both ct_0 and ct_1 means that now Alice only has to compute g^{r} once and send $g^{\mathsf{r}}, m_0 \cdot \mathsf{pk}_0^{\mathsf{r}}, m_1 \cdot \mathsf{pk}_1^{\mathsf{r}}$, meaning that it saves 1 exponentiation in terms

of computation and 1 group element in terms of communication. The same improvement is obtained when reusing randomness for computing $\mathsf{ct}'_0, \mathsf{ct}'_1$. Moreover, the fact that randomness is reused in generating $\mathsf{ct}'_0, \mathsf{ct}'_1$ makes it possible to improve the checks performed by Bob in step 3(b) of Protocol π_{OT}, saving 2 exponentiations. Namely, instead of computing both $\mathsf{Enc}(\mathsf{pk}_0, \widetilde{w}_0, \widetilde{\mathsf{r}}_0)$ and $\mathsf{Enc}(\mathsf{pk}_1, \widetilde{w}_1, \widetilde{\mathsf{r}}_1)$ to check whether the results match ct'_0 and ct'_1, it suffices to compute only $\mathsf{Enc}(\mathsf{pk}_{1-c}, \widetilde{w}_{1-c}, \widetilde{\mathsf{r}}_{1-c})$, check that $\widetilde{\mathsf{r}}_0 = \widetilde{\mathsf{r}}_1$ and check that $w_c = \widetilde{w}_c$. Notice that the two equality tests guarantee that w_c obtained from decrypting ct'_i is the same as the $\widetilde{w}_c$ contained in ch_c and that the randomness values $\widetilde{\mathsf{r}}_0$ and $\widetilde{\mathsf{r}}_1$ are also equal. Furthermore, computing $\mathsf{Enc}(\mathsf{pk}_{1-c}, \widetilde{w}_{1-c}, \widetilde{\mathsf{r}}_{1-c})$ and checking that it matches ct'_{1-c} guarantees that the first term of ciphertexts ct'_c and ct'_{1-c} is correctly computed, since $\mathsf{ct}'_i = (g^{\mathsf{r}'}, w_i \cdot \mathsf{pk}_i^{\mathsf{r}'})$ for $i \in \{0, 1\}$. Hence, computing $\mathsf{Enc}(\mathsf{pk}_c, \widetilde{w}_c, \widetilde{\mathsf{r}}_c)$ can only yield ct'_c, since PKE has Property 2.4, the same randomness r' is reused in both $\mathsf{ct}'_c, \mathsf{ct}'_{1-c}$ and $\widetilde{\mathsf{r}}_0 = \widetilde{\mathsf{r}}_1$. This optimized construction achieves a computational complexity of 6 exponentiations for the sender and 5 exponentiations for the receiver, while the total communication complexity is maintained at $2\lambda + 6\kappa + 7|\mathbb{G}|$ bits (where λ is message length, κ is the security parameter and $|\mathbb{G}|$ is the length of a group element).

5.5 Other Instantiations

In the full version of the paper we will present instantiations based on Learning with Errors (LWE), Quadratic Residuosity (QR), Decisional Composite Residuosity (DCR) and NTRU assumptions.

6 Implementation

Here, we propose concrete parameters and present implementation results for the QC-MDPC instantiation of our OT protocol. As the total cost of the entire OT protocol will be dominated by the key generation, encryption and decryption costs of the underlying public-key cryptosystem, we restrict our presentation to these respective costs.

6.1 Parameters

We computed the classical security level of a few parameter sets for QC-MDPC McEliece using a SAGE script. The quantum level is one half of the values here presented (e.g. to obtain 128-bit quantum secure parameters, take the 256-bit classically secure ones). The communication complexity is $2n$ bits for each party. Results are shown in table 6.1.

Security Level	Codimension r	Code length n	Row density w	Introduced errors t
256	32771	2*32771	2*137	264
192	19853	2*19853	2*103	199
128	10163	2*10163	2*71	134

Table 1: Parameters for QC-MDPC PKC

We recall that it is necessary that r be prime and that the polynomial $(x^r - 1)/(x - 1)$ be irreducible over GF(2). There exists a trade-off between communication costs and decoding complexity. For instance, these parameters are quite tight and decoding may be somewhat slow. By increasing r to, say, r = 49139 (50% more bandwidth), decoding speed roughly quadruples (encoding becomes about half as fast, but it is far faster than decoding to begin with).

6.2 Implementation

Some running times are shown in Table 2. These results are for a (non-vectorized) C implementation, compiled with 64-bit gcc under Windows 10, on an Intel i7-5500U @ 2.4 GHz with TurboBoost disabled. Running times are measured in kilocycles. The underlying PRNG is ChaCha20. The implementation is mostly isochronous, to prevent side-channel timing attacks.

Security Level	Key generation	Encryption n	Decryption
256	507.48 kcyc	483.51 kcyc	10,337.41 kcyc
128	101.59 kcyc	73.85 kcyc	1,758.66 kcyc
80	67.45 kcyc	52.49 kcyc	497.53 kcyc

Table 2: Running Times (in kilocycles) for QC-MDPC PKC.

7 Conclusions

In this work we presented a framework for obtaining efficient round-optimal UC-secure OT protocols that are secure against active adaptive adversaries. Our construction can be instantiated with the low noise LPN, McEliece, QC-MDPC, and CDH assumptions. Our instantiations based on the low noise LPN, McEliece, and QC-MDPC assumptions (which are more efficient than the previous works) are the first OT protocols, which are UC-secure against active adaptive adversaries based on these assumptions. We obtained the first CDH-based UC-secure protocol. In the full version, we will show that it is also possible to get instantiations based on LWE, QR, DCR and NTRU assumptions.

Acknowledgements

We would like to thank the anonymous reviewers of PKC 2018 for pointing out an issue in the security proof of an earlier version of this work.

References

[1] Rudolf Ahlswede and Imre Csiszár. On oblivious transfer capacity. In Harout Aydinian, Ferdinando Cicalese, and Christian Deppe, editors, *Information Theory, Combinatorics, and Search Theory*, volume 7777 of *Lecture Notes in Computer Science*, pages 145–166. Springer Berlin Heidelberg, 2013.

[2] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[3] Donald Beaver. Commodity-based cryptography (extended abstract). In *29th ACM STOC*, pages 446–455. ACM Press, May 1997.

[4] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 85–99. Springer, Heidelberg, January 2003.

[5] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990.

[6] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[7] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 65–86. Springer, Heidelberg, June 2015.

[8] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016.

[9] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 579–598. Springer, Heidelberg, July 2017.

[10] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, July 2003.

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

[12] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *39th FOCS*, pages 493–502. IEEE Computer Society Press, November 1998.

[13] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[14] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

[15] 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. ACM Press, May 2002.

[16] Ignacio Cascudo, Ivan Damgård, Bernardo Machado David, Irene Giacomelli, Jesper Buus Nielsen, and Roberto Trifiletti. Additively homomorphic UC commitments with optimal amortized overhead. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 495–515. Springer, Heidelberg, March / April 2015.

[17] 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, Heidelberg, March 2009.

[18] 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. Springer, Heidelberg, February / March 2013.

[19] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015.

[20] Claude Crépeau. Efficient cryptographic protocols based on noisy channels. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 306–317. Springer, Heidelberg, May 1997.

[21] Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *29th FOCS*, pages 42–52. IEEE Computer Society Press, October 1988.

[22] Claude Crépeau, Kirill Morozov, and Stefan Wolf. Efficient unconditional oblivious transfer from almost any noisy channel. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 47–59. Springer, Heidelberg, September 2005.

[23] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 247–264. Springer, Heidelberg, August 2003.

[24] 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. Springer, Heidelberg, December 2009.

[25] Ivan Damgård and Sunoo Park. How practical is public-key encryption based on LPN and ring-LPN? Cryptology ePrint Archive, Report 2012/699, 2012. `http://eprint.iacr.org/2012/699`.

[26] Bernardo David, Rafael Dowsley, and Anderson C. A. Nascimento. Universally composable oblivious transfer based on a variant of LPN. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 14*, volume 8813 of *LNCS*, pages 143–158. Springer, Heidelberg, October 2014.

[27] Bernardo Machado David and Anderson C. A. Nascimento. Efficient fully simulatable oblivious transfer from the mceliece assumptions. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 638–642. IEEE, 2011.

[28] Bernardo Machado David, Anderson C. A. Nascimento, and Rafael T. de Sousa Jr. Efficient fully simulatable oblivious transfer from the mceliece assumptions. *IEICE Transactions*, 95-A(11):2059–2066, 2012.

[29] 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 12*, volume 7412 of *LNCS*, pages 80–99. Springer, Heidelberg, August 2012.

[30] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 446–472. Springer, Heidelberg, February 2004.

[31] Nico Döttling, Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A CCA2 secure variant of the McEliece cryptosystem. *Information Theory, IEEE Transactions on*, 58(10):6672–6680, oct. 2012.

[32] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 164–181. Springer, Heidelberg, March 2011.

[33] 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. Springer, Heidelberg, December 2012.

[34] Rafael Dowsley, Felipe Lacerda, and Anderson C. A. Nascimento. Oblivious transfer in the bounded storage model with errors. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1623–1627, Honolulu, HI, USA, June 29 – July 4, 2014.

[35] Rafael Dowsley, Felipe Lacerda, and Anderson C. A. Nascimento. Commitment and oblivious transfer in the bounded storage model with errors. Cryptology ePrint Archive, Report 2015/952, 2015. `http://eprint.iacr.org/`.

[36] Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the possibility of universally composable commitments based on noisy channels. In André Luiz Moura dos Santos and Marinho Pilla Barcellos, editors, *Anais do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, SBSEG 2008*, pages 103–114, Gramado, Brazil, September 1–5, 2008. Sociedade Brasileira de Computação (SBC).

[37] 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. Springer, Heidelberg, April 2009.

[38] Rafael Dowsley, Jörn Müller-Quade, and Tobias Nilges. Weakening the isolation assumption of tamper-proof hardware tokens. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15*, volume 9063 of *LNCS*, pages 197–213. Springer, Heidelberg, May 2015.

[39] Rafael Dowsley and Anderson C. A. Nascimento. On the oblivious transfer capacity of generalized erasure channels against malicious adversaries: The case of low erasure probability. *IEEE Transactions on Information Theory*, 63(10):6819–6826, Oct 2017.

[40] 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, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 107–117. Springer, 2008.

[41] 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 08*, volume 5155 of *LNCS*, pages 107–117. Springer, Heidelberg, August 2008.

[42] 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.

[43] Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the composability of statistically secure bit commitments. *Journal of Internet Technology*, 14(3):509–516, 2013.

[44] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[45] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.

[46] 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. Springer, Heidelberg, May 2010.

[47] 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. Springer, Heidelberg, February 2004.

[48] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, Heidelberg, August 2009.

[49] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to encrypt with the LPN problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 679–690. Springer, Heidelberg, July 2008.

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

[51] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 52–66. Springer, Heidelberg, December 2001.

[52] 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. Springer, Heidelberg, May 2007.

[53] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 293–308. Springer, Heidelberg, August 2005.

[54] Preetha Mathew K., Sachin Vasant, Sridhar Venkatesan, and C. Pandu Rangan. A code-based 1-out-of-n oblivious transfer based on mceliece assumptions. In Mark D. Ryan, Ben Smyth, and Guilin Wang, editors, *Information Security Practice and Experience: 8th*

International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings, pages 144–157, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[55] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, Heidelberg, May 2007.

[56] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and concurrent security of the HB and HB+ protocols. *Journal of Cryptology*, 23(3):402–421, July 2010.

[57] Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd ACM STOC*, pages 316–324. ACM Press, May 2000.

[58] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, Heidelberg, May 2011.

[59] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 348–359. Springer, Heidelberg, September 2006.

[60] Julian Loss. Private communication.

[61] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *APPROX-RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.

[62] K. Preetha Mathew, Sachin Vasant, Sridhar Venkatesan, and C. Pandu Rangan. An efficient IND-CCA2 secure variant of the niederreiter encryption scheme in the standard model. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12*, volume 7372 of *LNCS*, pages 166–179. Springer, Heidelberg, July 2012.

[63] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN Progress Report 4244, Jet Propulsion Laboratory, 1978.

[64] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo SLM Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE, 2013.

[65] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[66] Anderson C. A. Nascimento and Andreas Winter. On the oblivious-transfer capacity of noisy resources. *Information Theory, IEEE Transactions on*, 54(6):2572–2581, June 2008.

[67] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15:159–166, 1986.

[68] 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.

[69] 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.

[70] 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. Springer, Heidelberg, August 2008.

[71] Adriana C. B. Pinto, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. Achieving oblivious transfer capacity of generalized erasure channels in the malicious model. *Information Theory, IEEE Transactions on*, 57(8):5566–5571, August 2011.

[72] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[73] Ronald L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivest- commitment.pdf, 1999.

[74] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.