Efficient Composable Oblivious Transfer from CDH in the Global Random Oracle Model

Bernardo David[1]* and Rafael Dowsley[2]

[1] IT University of Copenhagen
[2] Monash University

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 the first universally composable (UC) protocol for oblivious transfer secure against active static adversaries based on the Computational Diffie-Hellman (CDH) assumption. Our protocol is proven secure in the observable Global Random Oracle model. We start by constructing a protocol that realizes an OT functionality with a selective failure issue, but shown to be sufficient to instantiate efficient OT extension protocols. In terms of complexity, this protocol only requires the computation of 6 modular exponentiations and the communication of 5 group elements, five binary strings of security parameter length, and two binary strings of message length. Finally, we lift this weak construction to obtain a protocol that realizes the standard OT functionality (without any selective failures) at an additional cost of computing 9 modular exponentiations and communicating 4 group elements, four binary strings of security parameter length and two binary strings of message length. As an intermediate step before constructing our CDH based protocols, we design generic OT protocols from any OW-CPA secure public-key encryption scheme with certain properties, which could potentially be instantiated from more assumptions other than CDH.

1 Introduction

Oblivious transfer (OT) [39,28] is a fundamental cryptographic primitive that serves as a building block for a number of interesting applications, such as secure two-party and multi-party computation. In this work, we mainly focus on 1-out-of-2 string oblivious transfer, which is a two-party primitive. In this flavor of OT, the sender Alice inputs two strings m_0 and m_1, and the receiver Bob inputs a choice bit c, obtaining 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.

* This work was supported by a grant from Concordium Foundation and by Independent Research Fund Denmark grants number 9040-00399B (TrA^{2}C) and number 9131-00075B (PUMA).

The Universal Composability (UC) framework [8] is the most widely used methodology for analyzing protocol security under arbitrary composition. OT protocols UC-secure against static malicious adversaries can be designed under several computational assumptions, such as: Decisional Diffie-Hellman (DDH) [30,38], strong RSA [30], Quadractic Residuosity (QR) [38], Decisional Linear (DLIN) [34,17], Decisional Composite Residuosity (DCR) [34,14], McEliece Assumptions [19], low noise Learning Parity with Noise (LPN) [18] and Learning with Errors (LWE) [38]. Furthermore, there exist constructions based on simple generic primitives such as enhanced trapdoor functions [12] and public-key encryption plus semi-honest stand alone oblivious transfer [33], which mostly do not achieve the same efficiency as the constructions that leverage properties of specific computational assumptions.

It is a well-known fact that UC-secure OT protocols require a setup assumption [10]. Coincidentally, most of the UC-secure OT protocols (including the aforementioned ones) are based in the Common Reference String (CRS) model, where the parties are assumed to have access to a string randomly sampled from a given distribution before execution starts. While this setup assumption allows for the construction of efficient UC-secure OT protocols under a number of assumptions, questions have been raised about its practicality [11,15], since a local CRS is not readily available for a real world implementation of a protocol. Notice that OT can be UC-realized under a number of alternative setup assumptions, such as the public-key infrastructure model [16], the random oracle model (ROM) [4,6], noisy channels [24], tamper-proof hardware [35,23,25]. However, these models still require each instance of the protocol to access a local instance of the setup assumption. Informally, it means that each instance of the protocol uses an instance of the ideal functionality representing the setup assumption that is independent from all other instances and accessible only to the parties participating in the protocol execution but not to the environment.

Assuming that each protocol instance has local access to an independent setup in order to obtain secure composition is far from optimal and results in several issues that have been pointed out in previous works [9,5,11]. In particular, assuming the existence of independent random oracles (RO) for each protocol instance contradicts the common practice of replacing a random oracle by a standardized hash function, which is freely accessible and used by everybody. Such issues were first analyzed and addressed by Canetti *et al.* [9], who proposed the "Generalized UC model", where it is assumed that the instance of the trusted setup is globally available (and therefore also accessible by the environment) and used by all protocol instances. This formalism was subsequently extended to the random oracle setting by Canetti *et al.* [11], who define a global random oracle model, where a single instance of the random oracle $\mathcal{F}_{\mathrm{gRO}}$ is directly accessible by all parties, the adversary and the environment. Such a model precludes the use of proof techniques that require the simulator to "program" the random oracle's answers to a given query, which are usually employed in random oracle based constructions. UC protocols based on a local programmable CRS also suffer from issues similar to those of local programmable ROs [11], and formally the security guarantees for protocols based on local setups (e.g. local CRS or programmable RO) only hold if a new fresh setup is available for each individual instance of the protocol, which is unrealistic. It is not known how to generate even a single CRS without heuristics, let alone a fresh one for each execution. Quoting Canetti *et al.* [11] on the strength of the global random oracle model: "This model provides significantly stronger composable security guarantees than the traditional random oracle model of Bellare and Rogaway [4] or even the common reference string model". Note that more than one trusted setup instance can be available (in our construction we use 3 instances of global RO), but they should be globally available and not local for a protocol instance. Surprisingly, Canetti *et al.* [11] showed that using $\mathcal{F}_{\text{gRO}}$ as a setup assumption it is possible to construct universally composable DLOG based commitments and DDH based two-party computation and non-interactive secure computation secure against static malicious adversaries. Recently, new results in the global ROM were proven assuming certain relaxations of the model [7]. However, no efficient oblivious transfer protocol in the global random oracle model has been proposed so far.

1.1 Our contributions

We first propose a generic protocol for universally composable oblivious transfer secure against active static adversaries in the global random oracle model of [11]. The central building block of this construction is a One-Way Chosen Plaintext Attack (OW-CPA) secure public-key encryption (PKE) scheme with a number of properties. We show that such a scheme can be efficiently instantiated under the Computational Diffie Hellman (CDH) assumption. Our results can be summarized as follows:

– The *first* UC-secure OT protocol based on the CDH assumption. [3]

– The first UC-secure OT protocol in the Global Random Oracle model [11] that achieves efficiency for single executions (without OT extension) comparable to the most efficient previous work [38], which requires a programmable CRS.[4]

In order to obtain a protocol based on an assumption as weak as CDH, we introduce novel simulation techniques for extracting choice bits and messages in the simulation without resorting to programming the random oracle, which is not possible in the global random oracle model of Canetti *et al.* [11]. Notice that previous works required stronger computational assumptions (*e.g.* DDH [38,6]) even though they relied on stronger local setup assumptions (*e.g.* CRS [38] and programmable random oracles [6]). Hence, in comparison to such previous works, our results improve on both the computational and setup assumptions required for UC-secure OT.

[3] Döttling *et al.* [26] proposed an independent UC secure OT protocol in the CRS model with other techniques that yield CDH instantiations.

[4] The DDH based NISC of [11] is orders of magnitude less efficient than our approach and the protocol [13] has been introduced recently as independent work.

In terms of efficiency, our protocols compare favorably to previous works based on stronger assumptions. In the setting where one wishes to execute a large number of OTs through OT extension, the costs of each seed OT with our CDH based protocol are only the computation of 6 modular exponentiations and the communication of 5 group elements, 5 binary strings of security parameter length, and 2 binary strings of message length. In the setting where few OTs are needed, our CDH based protocol requires 15 modular exponentiations and the communication of 9 group elements, 9 binary strings of security parameter length, and 4 binary strings of message length. We remark that, in contrast to previous works based on local setup assumptions, our protocols can be readily implemented while retaining their security properties by substituting the global random oracle by an extensively tested cryptographic hash function (*e.g.* SHA3).

As an intermediate step towards our CDH based construction, we first design a generic protocol based on a public-key encryption scheme with certain properties. We start by constructing a generic protocol that realizes an OT functionality that captures a selective failure issue, which is nevertheless sufficient for instantiating efficient OT extension protocols as shown in [22]. Interestingly, our protocol achieves high efficiency, requiring only one key generation operation, two encryption operations and one 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, two ciphertexts, five binary strings of security parameter length, and two binary strings of message length. Later on, we obtain a generic protocol that realizes the standard OT functionality (without any selective failure) by augmenting our original protocol with four encryptions, one decryption, two ciphertexts, two binary strings of security parameter length and two binary strings of message length. If hundreds of OTs are needed, our OT with selective failures represents a new option of base OT for use with OT extension schemes. If only tens of OTs are needed, our OT without selective failures is a good option for usage. Besides yielding a CDH based instantiation, these generic protocols can be potentially instantiated under other assumptions, paving the way to post-quantum secure constructions of UC-secure OT under lattice and coding based assumptions.

1.2 Related Works

The global random oracle model has been established by Canetti *et al.* in [11], where they also build UC-secure commitments, two-party computation and non-interactive secure computation (NISC) secure against static malicious adversaries. In their construction of NISC in the global ROM, they state that a natural way to construct such a protocol would be to instantiate existing approaches based on 2-round OT with a global ROM version of the originally CRS based UC-secure OT protocol by Peikert *et al.* [38]. However, they observe that there are significant challenges in obtaining such a global ROM version of the protocol by Peikert *et al.*, and instead construct a one-side simulatable OT protocol that is only UC-secure against a malicious receiver. Their solution is *not generic* but intrinsically based on DDH via non-black-box use of the OT protocol of [38], only implying 2-round UC OT based on DDH, and with communication/computation costs several orders of magnitude higher than ours. On the other hand, ours is the first UC OT in the GRO built in a black-box way from a generic primitive (a PKE that we define), yielding the first UC OT based on CDH (a weaker assumption) while achieving much lower computation/communication costs. Even though the global ROM was recently revisited in [7], allowing for relaxations such as programming the random oracle in specific situations, no new results related to oblivious transfer were proposed in this relaxed model.

The idea of constructing OT using two public-keys — the "pre-computed" one and the "randomized" one dates back to early days of OT development [28,3]. Naor and Pinkas [37] presented an improved stand alone CDH-based protocol in the (local) random oracle model under the same approach that is proven secure in the half-simulation paradigm. A recent result by Friolo *et al.* [29] shows how to construct 4 round fully simulatable OT from key agreement protocols with certain properties without requiring setup assumptions, which yields a protocol based on CDH. However, their results fundamentally fall short of UC security (since UC-secure OT protocols necessarily require a setup assumption [10]) and cannot be easily adapted to this setting.

We remark that the "Simplest OT" protocol [15] and the protocol by Hauck and Loss [32] have been found to suffer from a number of issues [31,6] and are not UC secure. The CDH based protocol of [22] only realizes an OT functionality with a selective failure (as our first simple construction) and it is unclear how to use it to realize the standard OT functionality (without selective failure). The UC OT protocol of [1] can also be instantiated from a similar generic public key encryption scheme, for which a CDH instantiation is presented (among other assumptions). However, in order to prove the security of the construction of [1], it is also necessary to assume that the public key encryption scheme has circular security, which is an ad-hoc assumption not proven under CDH.

Independent and concurrent works: Döttling *et al.* [26] proposed a generic round optimal UC-secure OT protocol in the CRS model that can be instantiated from CDH. However, even though their protocol solves the important problem of achieving round optimality, it has computational and communication complexities orders of magnitude higher than our protocol, making it impractical. These overheads are intrinsic to the use of generic zero-knowledge proofs and garbled circuits in their construction. Canetti *et al.* [13] introduced a CDH based OT protocol that is UC-secure in the Global Random Oracle model. Similarly to our initial result, they focus on obtaining OT with selective failures in order to achieve better efficiency when using their protocol as basis for OT extension. However, differently from our final result, they do not show how to eliminate selective failures in their protocol without using OT extension.

1.3 Our Techniques

At a high level, we start by building a simple generic protocol that realizes a weak version of the OT functionality, which allows for a selective failure attack.

Starting from this weak flavor of OT is useful because it allows us to showcase our techniques more clearly while still being useful for performing OT extension, which results in an unlimited number of standard OTs (without selective failures) at very high efficiency. We then lift our protocol with selective failures to a generic protocol that realizes the standard OT functionality by leveraging subtleties of the first, simpler, construction. The central building block for both protocols is a public-key encryption (PKE) scheme satisfying a number of properties, which we construct based on the CDH assumption departing from the ElGamal cryptosystem. In order to provide some intuition on the design of our schemes, we informally describe properties we require from our PKE scheme and discuss how they are used to build our protocols:

- *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 a public/secret-key pair $(\mathsf{pk}_c, \mathsf{sk}_c)$ 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 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.
- *Property 4 (informal):* Property 3 only holds for key pairs generated according to the key generation algorithm or picked at random, but not for arbitrary key pairs, which could be crafted to be "non-committing". Intuitively, this property says that encrypting a message under such an arbitrary "non-committing" public key will also cause some message bits to be lost, which will come in handy in the security proof.
- *Property 5 (informal):* The PKE scheme has a witness-recovering decryption algorithm that outputs the randomness used to generate the decrypted ciphertext along with the plaintext message.

A Toy Example: Consider a very simple protocol where the receiver generates a key pair $(\mathsf{pk}_c, \mathsf{sk}_c)$, queries a global RO 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 recomputes pk_1 from pk_0 and s with the help of the RO and uses the public-keys to encrypt random seeds. The sender then uses these seeds to generate one-time pads (using the global RO) that she uses to encrypt her messages, sending both the PKE ciphertexts containing the seeds and the one-time pad encryptions of the actual messages to the receiver. The receiver can retrieve the seed encrypted under pk_c (since he has sk_c), compute the one-time pad with the help of the global RO and retrieve the message associated with his choice bit c. 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.

While this simple protocol intuitively implements a stand alone oblivious transfer, it is hard to construct a simulator to prove it UC-secure in the global RO model. If programming the RO was allowed, the simulator could program the answer of the RO to a query s in such a way that it knows the secret keys corresponding to both pk_0 and pk_1, allowing it to extract the messages from a corrupted sender. In the case of a corrupted receiver, the simulator could wait for the RO to be queried on one of the one-time pad seed (extracting the choice bit), retrieve the message associated to that choice bit and program the answer of this RO query in such a way that the one-time pad encryption related to that seed decrypts to the message obtained from the OT functionality. However, the global RO model precludes us from using any of these techniques. Instead, we develop novel techniques for extracting both a corrupted receiver's choice bit and a corrupted sender's messages solely by observing global RO queries.

OT with Selective Failures: As a starting point, we design a protocol that UC-realizes a weaker version of the OT functionality, which captures a selective failure attack. This attack allows a malicious sender to try and "guess" the receiver's choice bit, only being caught if her guess is wrong. Allowing this selective failure makes it easier to implement mechanisms used by the simulator to extract the choice bit from a malicious receiver without the need to program the random oracle. Even though this protocol has a selective failure issue, it has been shown in [22] that it is sufficient to instantiate efficient OT extension protocols such as the one of [36]. Many applications require such a high number of oblivious transfers that it makes sense to use an actual OT protocol only to seed an OT extension, which can then be used for an unlimited number of OTs at very low cost. In order to simulate an execution with a corrupted receiver, we augment our simple protocol with a challenge-response mechanism inspired by [22] that forces the receiver to query the global RO in such a way that it reveals its choice bit to the simulator. In the real world protocol, the adversary can mount a selective failure attack where it can "guess" the receiver's choice bit, being caught if it guesses the wrong bit. However, a simulator who can observe the queries made to the global RO can easily determine the receiver's choice bit without resorting to a selective failure attack. This mechanism works by having the sender pick two random values $\mathsf{p}_0, \mathsf{p}_1$, compute a challenge $\mathsf{ch} = \mathsf{H}(\mathsf{H}(\mathsf{p}_0)) + \mathsf{H}(\mathsf{H}(\mathsf{p}_1))$ where $\mathsf{H}(\cdot)$ is the random oracle and send this challenge to the receiver along with encryptions of $\mathsf{p}_0, \mathsf{p}_1$. The receiver decrypts p_c corresponding to its choice bit and answers with $\mathsf{chr} = \mathsf{H}(\mathsf{H}(\mathsf{p}_c)) + c \cdot \mathsf{ch}$, which will always be $\mathsf{H}(\mathsf{H}(\mathsf{p}_0))$ when ch is computed correctly. After receiving chr, the sender provides the receiver with $\mathsf{H}(\mathsf{p}_0)$ and $\mathsf{H}(\mathsf{p}_1)$, so that it can check that ch was correctly computed and that $\mathsf{H}(\mathsf{p}_c)$ is consistent with the value it decrypted. However, a malicious sender can always guess the receiver's choice bit and compute ch in such a way that it will learn the actual choice bit but only be caught if it guesses wrong. Due to Properties 1 and 3, the simulator can be assured that the query p_c done by the receiver corresponds to its choice bit. The case of a corrupted sender is handled by a novel technique where the sender is forced to query the global RO in a way that reveals both of its messages to a simulator who can observe RO queries. The basic idea is to modify the challenge-response mechanism by having the sender query the global RO not only with the challenge seed p_i but also adding the public-key pk_i, and randomness r_i used to encrypt p_i to the query. Using Property 5, the receiver can complete the challenge-response mechanism since it can recover r_i used in the encryption of p_i. Using Property 3, the simulator is assured that a malicious sender could only have generated one such query for each pair of value p_i and randomness r_i. Hence, the simulator can check which pairs $\mathsf{r}_i, \mathsf{p}_i$ in the list of queries to the global RO results in the ciphertexts sent by the sender when used as input to an encryption under pk_i. After extracting both $\mathsf{p}_0, \mathsf{p}_1$, the simulator detects whether the adversary is trying to guess the choice bit (as well as the bit being guessed), which it forwards to the functionality. Later on, the sender uses the same p_i and corresponding randomness r_i to query a different instance of the global RO and obtain a one-time pad used to encrypt the actual messages it wants to transfer. Hence, the simulator can also use $\mathsf{p}_0, \mathsf{p}_1$ to extract both messages transferred by a malicious sender.

Eliminating Selective Failures: We are also interested in solving the problem of directly UC-realizing a standard OT functionality in the observable global random oracle model. In order to do so, we must eliminate the selective failure issue of our first protocol. We observe that we can do so by basically running two instances of our first protocol in parallel with the same public-keys pk_0 and pk_1. Notice that these public-keys encode the choice bit, meaning that the same choice bit is used in both instances. The first instance will be used to extract the receiver's choice bit while ensuring a malicious sender cannot learn it through a selective failure attack. The other instance will be used to execute an oblivious transfer with the previously extracted choice bit and random messages, which can be later derandomized through standard techniques. We will run both protocol instances with a random choice bit, so that the receiver's actual choice bit does not leak in case the sender mounts a selective failure attack, which will be detected causing the execution to abort. In one of these instances, we will execute the challenge-response mechanism with the additional requirement that the sender must reveal both $\mathsf{p}_0, \mathsf{r}_0$ and $\mathsf{p}_1, \mathsf{r}_1$, allowing the receiver to be sure no selective failure attack occurred. With this instance we are able to extract the receiver's random choice bit while ensuring that in the second instance the same bit will be used (because it is encoded in the keys pk_0 and pk_1, also used in the second instance). In the second instance, we do not execute the challenge-response mechanism but use pk_0 and pk_1 to encrypt a second pair of seeds $\hat{\mathsf{p}}_0, \hat{\mathsf{p}}_1$ with randomness $\mathsf{r}'_0, \mathsf{r}'_1$, which the sender queries to another instance of the global RO to obtain one-time pads for random messages being transferred. Due to Property 3, the simulator can extract $\hat{\mathsf{p}}_0, \hat{\mathsf{p}}_1$ from the queries to the global RO and retrieve these random messages. At this point we have executed a random oblivious transfer, which is then derandomized to the receiver's actual choice bit and the sender's actual messages using standard information theoretical techniques.

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. When the coins r are specified we use $y \leftarrow F(x; r)$. 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 $\mathcal{Y}$. We will denote by $\mathsf{negl}(\kappa)$ the set of negligible functions of κ. We abbreviate *probabilistic polynomial time* as PPT.

Encryption Schemes: The main building block used in our OT protocol is a public-key encryption scheme PKE. It 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 a PPT key generation algorithm KG, a PPT encryption algorithm Enc and a deterministic decryption algorithm Dec. 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 used randomness.

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. Our OT protocol 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 1 (Double OW-CPA Security). 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:

$$
\begin{aligned}
&q \xleftarrow{\$} \mathcal{PK}\\
&(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(q) \text{ such that } \mathsf{pk}_0, \mathsf{pk}_1 \in \mathcal{PK} \text{ and } \mathsf{pk}_0 \star \mathsf{pk}_1 = q\\
&\mathsf{m}_i \xleftarrow{\$} \mathcal{M} \text{ for } i = 0, 1\\
&\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_i) \text{ for } i = 0, 1\\
&(\widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}) \xleftarrow{\$} \mathcal{A}_2(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{st})
\end{aligned}
$$

it holds that

$$\Pr[(\widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}) = (\mathsf{m}_0, \mathsf{m}_1)] \in \mathsf{negl}(\kappa).$$

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

Property 2 (Pseudorandomness of Public-Keys). 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 3 (Committing Encryption). Consider the public-key encryption scheme PKE and the security parameter κ. For every PPT adversary $\mathcal{A}$, it holds that:

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

Note that if Properties 2 and 3 hold for some PKE, then the modified version of Property 3 in which pk is chosen using KG also trivially holds. Moreover, we will need a variation of the committing property stating that even if an adversary is allowed to provide an arbitrary secret and public-key pair, it cannot both decrypt a ciphertext generated under that public-key *and* break the standard committing property. The rationale behind this property is that, for some committing encryption schemes, an adversary can generate an arbitrary public-key that breaks the standard committing property. However, in most cases, such a public-key will also cause plaintext information to be lost, making it impossible for the adversary to recover the original message from a ciphertext encrypted under this key with probability 1. This property is formalized in Property 4.

Property 4 (Committing Encryption with Arbitrary Keys). Consider the public-key encryption scheme PKE and the security parameter κ. For every PPT two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running the following experiment:

$(\mathsf{pk}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\kappa)$
$\mathsf{m} \xleftarrow{\$} \mathcal{M}$, $\mathsf{r} \xleftarrow{\$} \mathcal{R}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$
$((\mathsf{m}', \mathsf{r}'), (\mathsf{m}_1, \mathsf{r}_1), \ldots, (\mathsf{m}_{n-1}, \mathsf{r}_{n-1})) \xleftarrow{\$} \mathcal{A}_2(\mathsf{ct}, \mathsf{st})$

it holds that

$$\Pr[\mathsf{m}' = \mathsf{m} \wedge \mathsf{r}' = \mathsf{r} \wedge (\mathsf{m}_i, \mathsf{r}_i) \neq (\mathsf{m}, \mathsf{r}) \wedge \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i, \mathsf{r}_i)\ \forall\, i = 1, \ldots, n-1] \leq \frac{1}{n} + \mathsf{negl}(\kappa).$$

We require PKE to have a *witness-recovering* decryption algorithm. Informally, this property means that the decryption algorithm also recovers the randomness used to generate the ciphertext it takes as input. Witness-recovering decryption is formally defined in Property 5.

Property 5 (Witness-Recovering Decryption). A public-key encryption scheme $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ has a witness-recovering decryption algorithm Dec if it takes as input the secret-key $\mathsf{sk} \in \mathcal{SK}$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ and outputs either a pair (m, r) for $\mathsf{m} \in \mathcal{M}$ and $\mathsf{r} \in \mathcal{R}$ or an error symbol $\perp$. For any $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KG}(1^\kappa)$, any $\mathsf{m} \in \mathcal{M}$, any $\mathsf{r} \overset{\$}{\leftarrow} \mathcal{R}$ and $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$, it should hold that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = (\mathsf{m}, \mathsf{r})$ with overwhelming probability over the randomness used by the algorithms.

In Appendix D, we prove that Properties 1, 2, 3 and 4 hold for the ElGamal cryptosystems based on the CDH assumption, yielding an efficient instantiation of our generic protocol. Even though the ElGamal cryptosystem does not have a straightforward witness-recovery decryption algorithm, we show how any OW-CPA secure public-key encryption scheme used on random messages can be augmented with such a decryption algorithm to achieve Property 5. This can be done through the encrypt-with-hash paradigm, where the randomness used for encryption is obtained by hashing the message being encrypted, which can be proven secure in the non-programmable random oracle model.

Universal Composability in the Global Random Oracle Model: We analyze our protocol in the UC model with global random oracles as presented in [11]. We refer interested readers to the original work for more details on the UC framework [8]. In the UC model with global random oracles, the parties are assumed to have access to a global random oracle functionality $\mathcal{F}_{\mathrm{gRO}}$ (see Figure 1 for details) and interfaces that leak the list of illegitimate queries $\mathcal{Q}_{|s}$ to the adversary. Differently from the basic UC model, the global random oracle model allows all parties (including the environment) to access a single instance of $\mathcal{F}_{\mathrm{gRO}}$. The $\mathcal{F}_{\mathrm{gRO}}$ functionality functions as a regular random oracle but is augmented with a mechanism for leaking queries performed by parties that are not part of a given execution. In the UC model parties are identified by a unique pair of program id (PID) and session id (SID). Queries that are no prepended with the same SID as the one identifying the party $P = (\mathsf{pid}, \mathsf{sid})$ making the query are added to a list of illegitimate queries that can be requested by instances of functionalities whose session id match the one in the query. This mechanism allows the simulator to learn queries made by the environment or adversary but keeps the queries made by honest parties secret (as honest parties will follow the protocol and prepend their queries with the correct SID). Moreover, the functionalities in the global random oracle model take into consideration the existence of this list of illegitimate queries, requesting it from $\mathcal{F}_{\mathrm{gRO}}$ and handing it to the adversary, if requested by the adversary. Our construction will actually use three instances of $\mathcal{F}_{\mathrm{gRO}}$: $\mathcal{F}_{\mathrm{gRO1}}$ with range $\mathcal{PK}$, $\mathcal{F}_{\mathrm{gRO2}}$ with range $\{0,1\}^\lambda$ and $\mathcal{F}_{\mathrm{gRO3}}$ with range $\{0,1\}^\kappa$.

Functionality $\mathcal{F}_{\mathrm{gRO}}$

$\mathcal{F}_{\mathrm{gRO}}$ is parameterized by a range $\mathcal{D}$ and a list of ideal functionalities $\overline{\mathcal{F}}$.

- Upon receiving a query x from some party $P = (\mathsf{pid}, \mathsf{sid})$ or from the adversary $\mathcal{S}$ do:
  - If there is a pair (x, v) for some $v \in \mathcal{D}$ in the (initially empty) list $\mathcal{Q}$ of past queries, return v to P. Else, sample $v \xleftarrow{\$} \mathcal{D}$ and store the pair (x, v) in $\mathcal{Q}$. Return v to P.
  - Parse x as (s, x'). If $\mathsf{sid} \neq s$, then add (s, x', v) to the (initially empty) list of illegitimate queries for SID s, denoted by $\mathcal{Q}_{|s}$.
- Upon receiving a request from an instance of an ideal functionality in the list $\overline{\mathcal{F}}$, with SID s, return to this instance the list $\mathcal{Q}_{|s}$ of illegitimate queries for SID s.

Fig. 1. Functionality $\mathcal{F}_{\mathrm{gRO}}$.

We consider a static malicious adversary. I.e., it can deviate from the prescribed protocol in an arbitrary way, but has to corrupt the parties before the execution starts.

Oblivious Transfer: The functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ that provides ℓ instances of the 1-out-of-2 string (of length λ) oblivious transfer in the $\mathcal{F}_{\mathrm{gRO}}$-hybrid model is presented in Appendix A. This work focus on obtaining a weaker form of oblivious transfer that allows selective failure attacks, aiming for the same type of weaker OT as in Doerner et al. [22]. The ideal functionality $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ for 1-out-of-2 string oblivious transfer with selective failure in the $\mathcal{F}_{\mathrm{gRO}}$-hybrid model is described in Figure 2. Essentially, the sender is given the option of trying to guess the choice bit of the receiver. If she makes a wrong guess, the cheating is detected and the execution aborts. If she makes a right guess, she learns the choice bit and nothing is detected by the receiver. As proved by Doerner et al. in the full version of their work [21], $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ can be used as the base OTs in the OT extension protocol of Keller et al. [36] to UC-realize $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$.

Lemma 1. *The OT extension protocol of Keller et al. [36] UC-realizes $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ in the $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}, \mathcal{F}_{\mathrm{gRO}}$-hybrid model.*

Proof. This follows directly from Lemma D.3 of [21], which proves that the first part of the OT extension protocol UC-realizes the correlated OT with errors functionality $\mathcal{F}_{\mathrm{COTe}}$ in the $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}, \mathcal{F}_{\mathrm{gRO}}$-hybrid model, and the reduction from $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ to $\mathcal{F}_{\mathrm{COTe}}$ using the remaining steps of the OT extension protocol [36].

3 The Generic Protocol

Our protocol uses as a building block a public-key encryption scheme that satisfies Properties 1, 2, 3, 4 and 5 (defined in Section 2). The basic high-level idea is

Functionality $\mathcal{F}_{\text{SFOT}}^{\lambda}$.

$\mathcal{F}_{\text{SFOT}}^{\lambda}$ is parameterized by the length of the messages $\lambda \in \mathbb{N}$, which is publicly known. $\mathcal{F}_{\text{SFOT}}^{\lambda}$ interacts with a sender Alice and a receiver Bob, proceeding as follows:

- Upon receiving a message $(\mathsf{choose}, \mathsf{sid}, c)$ from Bob, where $c \in \{0,1\}$, record $(\mathsf{sid}, \mathsf{choice}, c)$, send $(\mathsf{chosen}, \mathsf{sid})$ to Alice, and ignore future messages $(\mathsf{choose}, \mathsf{sid}, \cdot)$ with the same sid.
- Upon receiving a message $(\mathsf{guess}, \mathsf{sid}, \hat{c})$ from Alice, where $\hat{c} \in \{0, 1, \perp, \mathsf{force}\}$, if a tuple $(\mathsf{sid}, \mathsf{choice}, c)$ is recorded, then record $(\mathsf{sid}, \mathsf{guess}, \hat{c})$, ignore future messages $(\mathsf{guess}, \mathsf{sid}, \cdot)$ with the same sid and do the following:
  1. If $\hat{c} = \perp$, send $(\mathsf{no-cheat}, \mathsf{sid})$ to Bob.
  2. If $\hat{c} = c$, send $(\mathsf{cheat-undetected}, \mathsf{sid})$ to Alice and $(\mathsf{no-cheat}, \mathsf{sid})$ to Bob.
  3. If $\hat{c} \neq c$ or $\hat{c} = \mathsf{force}$, send $(\mathsf{cheat-detected}, \mathsf{sid}, c)$ to both Alice and Bob.
- Upon receiving a message $(\mathsf{send}, \mathsf{sid}, \boldsymbol{x}_0, \boldsymbol{x}_1)$ from Alice, where each $\boldsymbol{x}_i \in \{0,1\}^{\lambda}$, if there are tuples $(\mathsf{sid}, \mathsf{choice}, c)$ and $(\mathsf{sid}, \mathsf{guess}, \hat{c})$ recorded such that $\hat{c} = \perp$ or $\hat{c} = c$, then send $(\mathsf{output}, \mathsf{sid}, \boldsymbol{x}_c)$ to Bob and ignore further messages from Alice with the same sid.
- When asked by $\mathcal{S}$, obtain from $\mathcal{F}_{\text{gRO}}$ the list $\mathcal{Q}_{|\mathsf{sid}}$ of illegitimate queries for SID sid and send it to $\mathcal{S}$.

Fig. 2. Functionality $\mathcal{F}_{\text{SFOT}}^{\lambda}$ in the Global Random Oracle model.

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 two 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 1 and uses the random oracle to force the element q to be chosen uniformly at random from $\mathcal{PK}$. After generating the pair of public and secret-key $(\mathsf{pk}_c, \mathsf{sk}_c)$, Bob samples a seed s, queries the random oracle $\mathcal{F}_{\text{gRO1}}$ with s 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, Alice learns nothing about Bob's choice bit. Next, Alice picks two uniformly random strings $\mathsf{p}_0, \mathsf{p}_1$, queries them to the random oracle $\mathcal{F}_{\text{gRO2}}$ obtaining $\widetilde{\mathsf{p}_0}, \widetilde{\mathsf{p}_1}$ as response, and then she computes one-time pad encryptions of her messages $\mathsf{m}_0, \mathsf{m}_1$ as $\widetilde{\mathsf{m}_0} = \mathsf{m}_0 \oplus \widetilde{\mathsf{p}_0}$ and $\widetilde{\mathsf{m}_1} = \mathsf{m}_1 \oplus \widetilde{\mathsf{p}_1}$. Alice also computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}_0, \mathsf{p}_0; \mathsf{r}_0)$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mathsf{p}_1; \mathsf{r}_1)$ and sends $(\widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{ct}_0, \mathsf{ct}_1)$ to Bob. Bob can use sk_c to decrypt ct_c obtaining p_c. He then queries p_c to the random oracle $\mathcal{F}_{\text{gRO2}}$ obtaining $\widetilde{\mathsf{p}_c}$ as response, and retrieves $\mathsf{m}_c = \widetilde{\mathsf{m}_c} \oplus \widetilde{\mathsf{p}_c}$. Due to Property 1, Bob will not be able to recover p_{1-c}

Protocol π_{SFOT}

Let PKE be a public-key encryption scheme that satisfies Properties 1, 2, 3, 4 and 5, and κ be the security parameter. Protocol π_{SFOT} is executed between Alice with inputs $\mathsf{m}_0, \mathsf{m}_1 \in \{0,1\}^\lambda$ and Bob with input $c \in \{0,1\}$. Three instances of the random oracle ideal functionality $\mathcal{F}_{\text{gRO}}$ are used: $\mathcal{F}_{\text{gRO1}}$ with range $\mathcal{PK}$, $\mathcal{F}_{\text{gRO2}}$ with range $\{0,1\}^\lambda$, and $\mathcal{F}_{\text{gRO3}}$ with range $\{0,1\}^\kappa$. Alice and Bob proceed as follows:

1. Bob generates a pair of keys $(\mathsf{pk}_c, \mathsf{sk}_c) \overset{\$}{\leftarrow} \mathsf{KG}(1^\kappa)$. He samples a random string $s \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and sends (sid, s) to $\mathcal{F}_{\text{gRO1}}$, obtaining q as answer. Bob computes pk_{1-c} such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$ and sends $(\mathsf{sid}, s, \mathsf{pk}_0)$ to Alice.
2. Upon receiving $(\mathsf{sid}, s, \mathsf{pk}_0)$ from Bob, Alice queries $\mathcal{F}_{\text{gRO1}}$ with (sid, s), obtaining answer q. Alice computes pk_1 such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$. She samples $\mathsf{p}_0, \mathsf{p}_1 \overset{\$}{\leftarrow} \{0,1\}^\kappa$, $\mathsf{r}_0, \mathsf{r}_1 \overset{\$}{\leftarrow} \mathcal{R}$ and queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_0, \mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{sid}, \mathsf{pk}_1, \mathsf{p}_1, \mathsf{r}_1)$, obtaining p'_0 and p'_1 as answers. She then queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{p}'_0)$ and $(\mathsf{sid}, \mathsf{p}'_1)$, obtaining p''_0 and p''_1 as answers, and computes $\mathsf{ch} \leftarrow \mathsf{p}''_0 \oplus \mathsf{p}''_1$. Alice computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pk}_0, \mathsf{p}_0; \mathsf{r}_0)$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mathsf{p}_1; \mathsf{r}_1)$, and sends $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ to Bob.
3. Upon receiving $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ from Alice, Bob computes $(\mathsf{p}_c, \mathsf{r}_c) \leftarrow \mathsf{Dec}(\mathsf{sk}_c, \mathsf{ct}_c)$. Bob queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ obtaining p'_c as the answer, and then with $(\mathsf{sid}, \mathsf{p}'_c)$ obtaining p''_c. Bob computes $\mathsf{chr} \leftarrow \mathsf{p}''_c \oplus (c \cdot \mathsf{ch})$ and sends $(\mathsf{sid}, \mathsf{chr})$ to Alice.
4. Upon receiving $(\mathsf{sid}, \mathsf{chr})$ from Bob, Alice verifies that $\mathsf{chr} = \mathsf{p}''_0$. If this check fails, Alice aborts. Otherwise, Alice queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_0, \mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{sid}, \mathsf{pk}_1, \mathsf{p}_1, \mathsf{r}_1)$, obtaining $\widetilde{\mathsf{p}_0}$ and $\widetilde{\mathsf{p}_1}$ as answers. Alice computes $\widetilde{\mathsf{m}_0} = \widetilde{\mathsf{p}_0} \oplus \mathsf{m}_0$, $\widetilde{\mathsf{m}_1} = \widetilde{\mathsf{p}_1} \oplus \mathsf{m}_1$. Alice sends $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ to Bob.
5. Upon receiving $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ from Alice, Bob checks if the p'_c that he received from Alice matches the one he locally computed. He also queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{p}'_{1-c})$ obtaining p''_{1-c}, and checks if $\mathsf{ch} = \mathsf{p}''_0 \oplus \mathsf{p}''_1$. If any check fails, Bob aborts. Otherwise, he queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ obtaining $\widetilde{\mathsf{p}_c}$ as answer, and computes $\mathsf{m}_c \leftarrow \widetilde{\mathsf{m}_c} \oplus \widetilde{\mathsf{p}_c}$. Bob outputs m_c.

Fig. 3. Protocol π_{SFOT}

in order to query it to the random oracle and to decrypt $\widetilde{\mathsf{m}_{1-c}}$. Therefore, the security for Alice is also guaranteed.

Even though this simple protocol seemingly performs an oblivious transfer, it poses significant challenges for a proof in the Global Random Oracle model of Canetti et al. [11], where the simulator cannot program the answers to random oracle queries. In the case of a malicious sender, the simulator would need to generate a seed s and public key pk_0 such that it knows both secret keys associated to the resulting public keys pk_0 and pk_1, which it needs to know in order to extract the messages m_0 and m_1. However, while this is easy if the simulator could program an arbitrary random oracle answer given the seed s, it cannot be done in this model. In the case of a malicious receiver, Property 2 ensures that the simulator cannot learn any information about the choice bit c before the adversary queries the random oracle on p_c, which only happens *after* the simulator has sent its last message. The simulator could possibly program the random oracle answer given p_c so that the result is m_c (received from the OT functionality), but this is not possible in this setting. In order to circumvent these challenges, we augment the simple protocol described before with mechanisms that allow the simulator to extract the choice bit c and messages m_0 and m_1 without resorting to programming the random oracle.

In order to obtain security against a malicious receiver, we use a challenge-response mechanism that follows the approach of Doerner et al. [22]. Basically, before carrying out the actual transfer, Alice queries $(\mathsf{pk}_0, \mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{pk}_1, \mathsf{p}_1, \mathsf{r}_1)$ to the random oracle $\mathcal{F}_{\mathrm{gRO3}}$ (note that this oracle is different from $\mathcal{F}_{\mathrm{gRO2}}$) obtaining $\mathsf{p}'_0, \mathsf{p}'_1$, and then queries $\mathsf{p}'_0, \mathsf{p}'_1$ to the random oracle $\mathcal{F}_{\mathrm{gRO3}}$ obtaining $\mathsf{p}''_0, \mathsf{p}''_1$. Alice fixes the challenge as $\mathsf{ch} \leftarrow \mathsf{p}''_0 \oplus \mathsf{p}''_1$ and sends ch to Bob. Bob queries $\mathcal{F}_{\mathrm{gRO3}}$ with $(\mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$, which is possible because PKE has witness-recovering decryption according to Property 5, obtaining p'_c and then with p'_c obtaining p''_c. Bob returns $\mathsf{p}''_c \oplus (c \cdot \mathsf{ch})$ to Alice, who checks if the returned value is equal to p''_0. Alice then sends $\mathsf{p}'_0, \mathsf{p}'_1$ to Bob, who checks if these values are compatible with the values he previously computed and ch. After receiving a valid response from Bob, Alice proceeds with the transfer. A crucial aspect of this mechanism is that in order to obtain p''_c, Bob is forced to first issue a query associated to its choice bit c to the random oracle, allowing for extraction. In the proof, the simulator can extract c solely by observing the adversary's queries after it receives the challenge, allowing it to obtain m_c from the OT functionality and prepare the last message to the adversary accordingly. This mechanism allows selective failure attacks, but the resulting scheme fulfills the requirements to be used as base OTs in the OT extension scheme of Keller et al. [36] (see Section 2).

Instead of querying $\mathcal{F}_{\mathrm{gRO2}}$ with $\mathsf{p}_0, \mathsf{p}_1$, we query it with $(\mathsf{pk}_0, \mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{pk}_1, \mathsf{p}_1, \mathsf{r}_1)$ to obtain $\widetilde{\mathsf{p}_0}, \widetilde{\mathsf{p}_1}$. These queries of the form $(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\mathrm{gRO2}}$ and $\mathcal{F}_{\mathrm{gRO3}}$ allow the simulator to extract both of the corrupt sender's messages solely by observing the queries to the random oracle. In the simulation, the simulator reconstructs ciphertexts $\hat{\mathsf{ct}}_j = \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_j, \hat{\mathsf{r}}_j)$ from all random oracle queries of the form $(\mathsf{pk}_i, \hat{\mathsf{p}}_j, \hat{\mathsf{r}}_j)$, looking for a ciphertext $\hat{\mathsf{ct}}_j$ that matches ciphertext ct_i (for $i \in \{0,1\}$) in the adversary's message. Having found these ciphertexts the simulator can proceed to recover each message m_i. An adversary could try to confuse the simulator by making two different queries to the random oracle that pass the tests above. However, this is not possible due to Properties 3 and 4.

Protocol π_{SFOT} is described in Figure 3 and its security if formally stated in Theorem 1, which we prove in Appendix B. A CDH based instantiation is described in Appendix D.

Theorem 1. *Let* PKE *be a public-key encryption scheme that satisfies Properties 1, 2, 3, 4 and 5. When instantiated with* PKE*, Protocol* π_{SFOT} *UC-realizes* *functionality $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ with security against static malicious adversaries in the global random oracle model.*

4 Realizing $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ directly

Our previous generic protocol can be modified to directly realize the standard 1-out-of-2 OT functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ without any selective failure issues, instead of first realizing $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ and then employing the OT extension of Keller *et al.* to realize $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$. However, we will rely directly on the specific CDH based PKE of Appendix D instead of a generic PKE with Properties 1, 2, 3, 4 and 5. This is necessary since the simulator will now need to extract messages encrypted under this PKE that it cannot extract by simply observing queries to the random oracle instances used in the protocol but that can be extracted by observing queries to the random oracle instance used by this specific PKE construction.

In order to eliminate the potential selective failure from our first protocol, we need to provide Bob with a proof that Alice has used exactly the values $\mathsf{p}_0, \mathsf{p}_1$ contained in ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ to generate challenge ch. The main idea is to use two instances of our original protocol that are run using the same public keys $\mathsf{pk}_0, \mathsf{pk}_1$ (encoding the same choice bit). One of them is used to execute the challenge-response mechanism and the other is used to execute a random OT, which can be later derandomized. In our previous protocol, Alice only reveals the outputs of $\mathcal{F}_{\mathrm{gRO3}}$ upon being queried with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$, which only allows Bob to check that these were the values used in the challenge with probability $\frac{1}{2}$. In order to prove that those values were indeed used, we will leverage the committing property (Property 3) of the underlying cryptosystem and have Alice reveal $\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1$ to Bob upon getting a valid response to the challenge. Using these values, Bob can recompute the challenge (checking that it matches ch received from Alice) and check that $\mathsf{ct}_i \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, for $i = \{0, 1\}$. If those checks fail, the receiver aborts but, if they succeed, it is assured by the committing property that those values were used in computing ct_0, ct_1 and ch (meaning the choice bit was not leaked). Having both $\mathsf{p}_0, \mathsf{p}_1$ revealed to Bob, we will need to have Alice generate new $\hat{\mathsf{p}}_0, \hat{\mathsf{p}}_1$ and corresponding $\hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1$ to complete the OT as in our first protocol. However, notice that this protocol still leaks Bob's choice bit to an adversary who mounts a successful selective failure attack, even though the attack is detected and the protocol is aborted. In order to deal with this, Bob uses a random choice bit to execute a random OT that is derandomized after Bob is certain no selective failure attack occurred.

The simulator for a corrupt Alice does not have to extract the "guess" bit of the adversary, just acting as an honest Bob and extracting the messages $\mathsf{m}_0, \mathsf{m}_1$ using the same techniques as the simulator in π_{SFOT}. However, it will need to extract messages $\hat{\mathsf{p}}_0, \hat{\mathsf{p}}_1$ from the ciphertexts $\hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1$ by observing queries to the random oracle used in the CDH based PKE from Appendix D. The simulator for a corrupt Bob uses the same techniques as the simulator in π_{SFOT} to extract the choice bit. The difference is that the ciphertexts $\mathsf{ct}'_0, \mathsf{ct}'_1$ obtained from the challenger in the game of Property 1 are given to the adversary as $\mathsf{ct}_c, \hat{\mathsf{ct}}_{1-c}$ in the reduction showing that an adversary that obtains m_{1-c} when interacting with this simulator breaks Property 1.

Protocol π_{OT} is described in Figure 4 and its security if formally stated in Theorem 2, which we prove in Appendix C. The CDH based PKE instantiation is described in Appendix D.

Theorem 2. *Under the CDH assumption, Protocol π_{OT} UC-realizes functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ with security against static malicious adversaries in the global random oracle model.*

References

1. Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the rom. Cryptology ePrint Archive, Report 2017/993, 2017. `https://eprint.iacr.org/2017/993`.
2. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, August 2007.
3. 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.
4. 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.
5. Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 51–70. Springer, Heidelberg, August 2011.
6. 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`.
7. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.
9. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.
10. 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.

Protocol π_{OT}

Let PKE be the CDH based public-key encryption scheme of Appendix D that satisfies Properties 1, 2, 3, 4 and 5, and κ be the security parameter. Protocol π_{OT} is executed between Alice with inputs $\mathsf{m}_0, \mathsf{m}_1 \in \{0,1\}^\lambda$ and Bob with input $c \in \{0,1\}$. Bob and Alice interact with each other and with four instances of the random oracle ideal functionality: $\mathcal{F}_{\text{gRO1}}$ with range $\mathcal{PK}$, $\mathcal{F}_{\text{gRO2}}$ with range $\{0,1\}^\lambda$, $\mathcal{F}_{\text{gRO3}}$ with range $\{0,1\}^\kappa$, and $\mathcal{F}_{\text{gRO4}}$ with range $\mathcal{R}$ (used by PKE). Protocol π_{OT} proceeds as follows:

1. Bob samples $c' \xleftarrow{\$} \{0,1\}$ and 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 $\mathcal{F}_{\text{gRO1}}$, obtaining q as answer. Bob computes $\mathsf{pk}_{1-c'}$ such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$ and sends $(\mathsf{sid}, s, \mathsf{pk}_0)$ to Alice.

2. Upon receiving $(\mathsf{sid}, s, \mathsf{pk}_0)$ from Bob, Alice queries $\mathcal{F}_{\text{gRO1}}$ with (sid, s), obtaining answer q and computing pk_1 such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$. For $i \in \{0,1\}$, Alice samples $\mathsf{p}_i, \hat{\mathsf{p}}_i \xleftarrow{\$} \{0,1\}^\kappa$ and $\mathsf{r}_i, \hat{\mathsf{r}}_i \xleftarrow{\$} \mathcal{R}$, queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$, obtaining p'_i as answer, and then queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{p}'_i)$, obtaining p''_i as answer. Alice computes $\mathsf{ch} \leftarrow \mathsf{p}''_0 \oplus \mathsf{p}''_1$. For $i \in \{0,1\}$, Alice computes $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_i; \hat{\mathsf{r}}_i)$. Alice sends $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1, \hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1)$ to Bob.

3. Upon receiving $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1, \hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1)$ from Alice, Bob computes $(\mathsf{p}_{c'}, \mathsf{r}_{c'}) \leftarrow \mathsf{Dec}(\mathsf{sk}_{c'}, \mathsf{ct}_{c'})$, queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$, obtaining $\mathsf{p}'_{c'}$ as the answer, and then queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{p}'_{c'})$, obtaining $\mathsf{p}''_{c'}$ as the answer. Bob computes $\mathsf{chr} \leftarrow \mathsf{p}''_{c'} \oplus (c' \cdot \mathsf{ch})$ and sends $(\mathsf{sid}, \mathsf{chr})$ to Alice.

4. Upon receiving $(\mathsf{sid}, \mathsf{chr})$ from Bob, Alice verifies that $\mathsf{chr} = \mathsf{p}''_0$. If this check fails, Alice aborts. Otherwise, for $i \in \{0,1\}$, Alice queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i, \hat{\mathsf{r}}_i)$ (obtaining $\widetilde{\mathsf{p}_i}$ as answer), samples $\hat{\mathsf{m}}_i \xleftarrow{\$} \{0,1\}^\lambda$ and computes $\widetilde{\mathsf{m}_i} = \widetilde{\mathsf{p}_i} \oplus \hat{\mathsf{m}}_i$. Alice sends $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ to Bob.

5. Upon receiving $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ from Alice, for $i \in \{0,1\}$, Bob checks that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ and queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ (obtaining p'_i as answer) and queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{sid}, \mathsf{p}'_i)$ (obtaining p''_i as answers). Next Bob checks that $\mathsf{ch} = \mathsf{p}''_0 \oplus \mathsf{p}''_1$. If any checks fail, Bob aborts. Otherwise, Bob computes $(\hat{\mathsf{p}}_{c'}, \hat{\mathsf{r}}_{c'}) \leftarrow \mathsf{Dec}(\mathsf{sk}_{c'}, \hat{\mathsf{ct}}_{c'})$. If this decryption fails with $\perp \leftarrow \mathsf{Dec}(\mathsf{sk}_{c'}, \hat{\mathsf{ct}}_{c'})$, Bob samples a random $(\hat{\mathsf{p}}_{c'}, \hat{\mathsf{r}}_{c'}) \xleftarrow{\$} \{0,1\}^\kappa \times \mathcal{R}$ in order to avoid a selective failure. Bob queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_{c'}, \hat{\mathsf{p}}_{c'}, \hat{\mathsf{r}}_{c'})$, obtaining $\widetilde{\mathsf{p}}_{c'}$ as answer, computes $\hat{\mathsf{m}}_{c'} \leftarrow \widetilde{\mathsf{m}_{c'}} \oplus \widetilde{\mathsf{p}_{c'}}$, sets $d \leftarrow c \oplus c'$ and sends (sid, d) to Alice.

6. Upon receiving (sid, d) from Bob, Alice sets $\mathsf{m}'_0 \leftarrow \hat{\mathsf{m}}_d \oplus \mathsf{m}_0, \mathsf{m}'_1 \leftarrow \hat{\mathsf{m}}_{1-d} \oplus \mathsf{m}_1$. Alice sends $(\mathsf{sid}, \mathsf{m}'_0, \mathsf{m}'_1)$ to Bob.

7. Upon receiving $(\mathsf{sid}, \mathsf{m}'_0, \mathsf{m}'_1)$ from Alice, Bob computes $\mathsf{m}_c = \mathsf{m}'_c \oplus \hat{\mathsf{m}}_{c'}$, outputs m_c and halts.

Fig. 4. Protocol π_{OT}

11. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 597–608. ACM Press, November 2014.
12. 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.
13. Ran Canetti, Pratik Sarkar, and Xiao Wang. Blazing fast OT for three-round UC OT extension. *IACR Cryptology ePrint Archive*, 2020:110, 2020. To appear at PKC 2020.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. Yevgeniy Dodis, Victor Shoup, and Shabsi Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 515–535. Springer, Heidelberg, August 2008.
21. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Secure two-party threshold ECDSA from ECDSA assumptions. Cryptology ePrint Archive, Report 2018/499, 2018. `https://eprint.iacr.org/2018/499`.
22. Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pages 980–997. IEEE Computer Society Press, May 2018.
23. 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.
24. Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the composability of statistically secure random oblivious transfer. *Entropy*, 22(1):107, 2020.

25. 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.
26. Nico Dttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from cdh or lpn. Cryptology ePrint Archive, Report 2019/414, 2019. `https://eprint.iacr.org/2019/414`.
27. 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.
28. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.
29. Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. Cryptology ePrint Archive, Report 2018/473, 2018. (To appear in TCC 2019) `https://eprint.iacr.org/2018/473`.
30. 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.
31. Ziya Alper Gen, Vincenzo Iovino, and Alfredo Rial. "the simplest protocol for oblivious transfer" revisited. Cryptology ePrint Archive, Report 2017/370, 2017. `https://eprint.iacr.org/2017/370`.
32. 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`.
33. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 183–209. Springer, Heidelberg, November / December 2015.
34. 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.
35. 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.
36. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.
37. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.
38. 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.
39. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

A OT Functionality

The functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ that provides ℓ instances of the 1-out-of-2 string (of length λ) oblivious transfer in the $\mathcal{F}_{\mathrm{gRO}}$-hybrid model is presented in Figure 5.

Functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$.

$\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ is parameterized by the length of the messages $\lambda \in \mathbb{N}$ and by the number of message pairs ℓ, which are publicly known. $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ interacts with a sender Alice and a receiver Bob, proceeding as follows:

– Upon receiving a message $(\mathsf{send}, \mathsf{sid}, \boldsymbol{x}_{0,1}, \boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{0,\ell}, \boldsymbol{x}_{1,\ell})$ from Alice, where each $\boldsymbol{x}_{i,j} \in \{0,1\}^\lambda$, store the tuple $(\mathsf{sid}, \mathsf{sent}, \boldsymbol{x}_{0,1}, \boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{0,\ell}, \boldsymbol{x}_{1,\ell})$ and send $(\mathsf{sent}, \mathsf{sid})$ to Bob. Ignore further messages from Alice with the same sid.

– Upon receiving a message $(\mathsf{choose}, \mathsf{sid}, c_1, \ldots, c_\ell)$ from Bob, where each $c_j \in \{0,1\}$, check if a tuple $(\mathsf{sid}, \mathsf{sent}, \boldsymbol{x}_{0,1}, \boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{0,\ell}, \boldsymbol{x}_{1,\ell})$ was recorded. If yes, send $(\mathsf{output}, \mathsf{sid}, \boldsymbol{x}_{c_1,1}, \ldots, \boldsymbol{x}_{c_\ell,\ell})$ to Bob and $(\mathsf{received}, \mathsf{sid})$ to Alice, and ignore further messages from Bob with the same sid. Otherwise, send nothing, but continue running.

– When asked by $\mathcal{S}$, obtain from $\mathcal{F}_{\mathrm{gRO}}$ the list $\mathcal{Q}_{|\mathsf{sid}}$ of illegitimate queries for SID sid and send it to $\mathcal{S}$.

Fig. 5. Functionality $\mathcal{F}_{\mathrm{OT}}^{\lambda,\ell}$ in the Global Random Oracle model.

B Security Analysis of Protocol π_{SFOT}

In this appendix we analyse the security of Protocol π_{SFOT} and present a full proof of Theorem 1. First we discuss the correctness of Protocol π_{SFOT}. In Steps 2 and 3, if the message $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ is correctly generated by Alice, Bob is able to decrypt ct_c with the secret-key sk_c to obtain p_c and r_c (due to Property 5), allowing him to compute the right answer to the challenge, and also to decrypt m_c in Step 5. We now formally state the security of π_{SFOT} in Theorem 1.

Theorem 1 *Let* PKE *be a public-key encryption scheme that satisfies Properties 1, 2, 3, 4 and 5. When instantiated with* PKE*, Protocol* π_{SFOT} *UC-realizes the functionality* $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ *against static malicious adversaries in the global random oracle model.*

Proof. In order to prove the security of π_{SFOT}, we will construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between interactions with an adversary $\mathcal{A}$ through π_{SFOT} in the real world and with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ in the ideal world. For the sake of clarity, we will describe the simulator $\mathcal{S}$ separately for 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 $\mathcal{Z}$. Notice that simulating the cases where both Bob and Alice are corrupted or honest is trivial. If both parties are corrupted, $\mathcal{S}$ simply runs $\mathcal{A}$ internally. In this case, $\mathcal{A}$ generates 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 $\mathcal{A}$. We analyze the cases where only Bob is corrupted and where only Alice is corrupted below.

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 $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Bob, proceeding as follows:

1. $\mathcal{S}$ forwards all messages between $\mathcal{A}$ and global random oracles $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$ and $\mathcal{F}_{\mathrm{gRO3}}$. Moreover, it keeps up-to-date lists of adversarial and illegitimate queries (and associated answers) for each global random oracle.
2. Upon receiving $(\mathsf{sid}, s, \mathsf{pk}_0)$ from $\mathcal{A}$, $\mathcal{S}$ follows the instructions of an honest Alice in Step 2 of π_{SFOT} to generate and send $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ to $\mathcal{A}$:
   (a) $\mathcal{S}$ sends (sid, s) to $\mathcal{F}_{\mathrm{gRO1}}$, receiving q as answer. $\mathcal{S}$ computes pk_1 such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$.
   (b) For $i \in \{0,1\}$, $\mathcal{S}$ samples $\mathsf{p}_i \xleftarrow{\$} \{0,1\}^{\kappa}$, $\mathsf{r}_i, \xleftarrow{\$} \mathcal{R}$, computes $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, and queries $\mathcal{F}_{\mathrm{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to obtain p'_i, and with $(\mathsf{sid}, \mathsf{p}'_i)$ to obtain p''_i.
   (c) $\mathcal{S}$ computes $\mathsf{ch} \leftarrow \mathsf{p}''_0 \oplus \mathsf{p}''_1$ and sends $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ to $\mathcal{A}$.
3. Whenever the first query $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathrm{gRO2}}$ or $\mathcal{F}_{\mathrm{gRO3}}$ where $c \in \{0,1\}$ happens, $\mathcal{S}$ sends $(\mathsf{choose}, \mathsf{sid}, c)$ to $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$. Upon receiving $(\mathsf{output}, \mathsf{sid}, \mathsf{m}_c)$ from $\mathcal{F}_{\mathrm{SFOT}}^{\lambda}$, $\mathcal{S}$ stores m_c.
4. Upon receiving $(\mathsf{sid}, \mathsf{chr})$ from $\mathcal{A}$, $\mathcal{S}$ checks that $\mathsf{chr} = \mathsf{p}''_0$, aborting otherwise (as an honest Alice would). If no query of the form $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ has been recorded in the lists of $\mathcal{F}_{\mathrm{gRO3}}$ or $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{S}$ outputs fail and halts. Otherwise, $\mathcal{S}$ samples $\mathsf{m}_{1-c} \xleftarrow{\$} \{0,1\}^{\lambda}$ and executes Step 4 of Protocol π_{SFOT} as an honest Alice to generate and send $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ to $\mathcal{A}$:
   (a) For $i \in \{0,1\}$, $\mathcal{S}$ queries $\mathcal{F}_{\mathrm{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ obtaining $\widetilde{\mathsf{p}_i}$ and computes $\widetilde{\mathsf{m}_i} = \widetilde{\mathsf{p}_i} \oplus \mathsf{m}_i$.
   (b) $\mathcal{S}$ sends $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ to $\mathcal{A}$.
5. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

Fig. 6. Simulator $\mathcal{S}$ for the case where only Bob is corrupted.

Simulator for the case only Bob *is corrupted.* In the case where only Bob is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}^{\lambda}_{\text{SFOT}}$ 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 receives queries from $\mathcal{A}$ to the instances of $\mathcal{F}_{\text{gRO}}$ (*i.e.* $\mathcal{F}_{\text{gRO1}}$, $\mathcal{F}_{\text{gRO2}}$ and $\mathcal{F}_{\text{gRO3}}$), which it forwards. When the query is answered, $\mathcal{S}$ forwards the answer to $\mathcal{A}$. The goal of the simulator is to extract the corrupted receiver's choice bit c in order to request the correct message from $\mathcal{F}^{\lambda}_{\text{SFOT}}$, which is later transferred to the adversary. The simulator $\mathcal{S}$ is presented in Figure 6. Notice that, unless $\mathcal{S}$ outputs fail, it executes all the steps of an honest Alice in the real protocol with the sole difference that it uses a random message $\mathsf{m}_{1-c} \xleftarrow{\$} \{0,1\}^{\lambda}$ instead of the real message. We will show that $\mathcal{S}$ only outputs fail with negligible probability and that simulation with a uniformly random m_{1-c} is indistinguishable from a real execution.

First, notice that $\mathcal{S}$ only outputs fail if it receives a message $(\mathsf{sid}, \mathsf{chr})$ from $\mathcal{A}$ such that $\mathsf{chr} = \mathsf{p}''_0$ without a query $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$, where $c \in \{0,1\}$, being made to $\mathcal{F}_{\text{gRO3}}$ or $\mathcal{F}_{\text{gRO2}}$ beforehand, meaning that $\mathcal{S}$ cannot extract the choice bit c. We remark that, in this argument, c is defined as the bit corresponding to the p_i value contained in the first such query and may not correspond to the actual choice bit. At this stage we are only interested in showing that $\mathcal{A}$ must issue a query $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\text{gRO3}}$ in order to obtain the correct chr, resulting in $\mathcal{S}$ proceeding without outputting fail. Later on, we will show that the extracted bit c is indeed $\mathcal{A}$'s choice bit, or rather that $\mathcal{A}$ cannot obtain m_{1-c} without violating one of the properties of PKE.

Notice that $\mathsf{ch} = \mathsf{p}''_0 \oplus \mathsf{p}''_1$ (where p''_0 and p''_1 are outputs of a random oracle) and that the adversary $\mathcal{A}$ only receives $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ up to the challenge phase of the protocol. Hence, in order to obtain p''_0 and pass the challenge-response test with non-negligible probability, $\mathcal{A}$ needs to: either (1) query $\mathcal{F}_{\text{gRO3}}$ on $(\mathsf{sid}, \mathsf{pk}_0, \mathsf{p}_0, \mathsf{r}_0)$ to obtain p'_0 and then p''_0; or (2) query $\mathcal{F}_{\text{gRO3}}$ on $(\mathsf{sid}, \mathsf{pk}_1, \mathsf{p}_1, \mathsf{r}_1)$ to obtain p'_1 followed by p''_1 and then compute $\mathsf{p}''_0 \leftarrow \mathsf{ch} \oplus \mathsf{p}''_1$.

Given that $\mathcal{S}$ does not output fail, it extracts a choice bit c and obtains the message m_c from $\mathcal{F}^{\lambda}_{\text{SFOT}}$. The only difference between the simulation and a real execution is that $\mathcal{S}$ computes $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ using a random message $\mathsf{m}_{1-c} \leftarrow \{0,1\}^{\lambda}$ instead of the real message. Notice that m_{1-c} is "one-time pad encrypted" in $\widetilde{\mathsf{m}_{1-c}} = \widetilde{\mathsf{p}_{1-c}} \oplus \mathsf{m}_{1-c}$. Hence, $\mathcal{A}$ can only learn any information about m_{1-c} with non-negligible probability if it queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{pk}_{1-c}, \mathsf{p}_{1-c}, \mathsf{r}_{1-c})$ to obtain $\widetilde{\mathsf{p}_{1-c}}$. We will show that even though m_{1-c} is chosen uniformly at random, the simulation is indistinguishable from the real execution because $\mathcal{A}$ cannot learn anything about m_{1-c} without breaking some property of PKE. The main idea is to show that we can build an adversary $\mathcal{A}_1$ that breaks Property 1 with probability p given black-box access to a pair of $\mathcal{A}$ and $\mathcal{Z}$ such that $\mathcal{Z}$ distinguishes the ideal execution with $\mathcal{F}^{\lambda}_{\text{SFOT}}$ and $\mathcal{S}$ from a real execution with $\mathcal{A}$ and π_{SFOT} with probability p, *i.e.* where $\mathcal{A}$ queries $\mathcal{F}_{\text{gRO3}}$ with $(\mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ and later manages to query $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{pk}_{1-c}, \mathsf{p}_{1-c}, \mathsf{r}_{1-c})$ also with probability p.

Reduction from a pair $(\mathcal{Z}, \mathcal{A})$ for which simulation fails to $\mathcal{A}_1$ that breaks Property 1: Given a pair of $\mathcal{A}$ and $\mathcal{Z}$ such that $\mathcal{Z}$ distinguishes the ideal execution with $\mathcal{F}_{\mathsf{SFOT}}^{\lambda}$ and $\mathcal{S}$ from a real execution with $\mathcal{A}$ and π_{SFOT} with probability p, we construct an adversary $\mathcal{A}_1$ that breaks Property 2 with probability at least p. $\mathcal{A}_1$ interacts with the challenger in the game of Property 1 and with *copies* of $\mathcal{A}$ and $\mathcal{Z}$, for which it simulates $\mathcal{S}$, $\mathcal{F}_{\text{gRO1}}$, $\mathcal{F}_{\text{gRO2}}$ and $\mathcal{F}_{\text{gRO3}}$. $\mathcal{A}_1$ first receives q from the challenger in the game of Property 1, then randomly picks a query (sid, s) from the set of $\mathcal{A}$'s queries to $\mathcal{F}_{\text{gRO1}}$ and answers it with (sid, q), where q was received from the challenger of the security game. We remark that the no programming is done by $\mathcal{S}$ in the simulation of π_{SFOT}. Notice that $\mathcal{A}_1$ programs the random oracle emulated inside this reduction where *copies* of $\mathcal{Z}$ and $\mathcal{A}$ are used in a black-box way to break Property 1. In this reduction, $\mathcal{A}_1$ (but *not* $\mathcal{S}$) uses copies of both $\mathcal{Z}$ and $\mathcal{A}$ and emulates all the ideal functionalities and parties towards these copies. $\mathcal{A}_1$ does not interact with the actual environment $\mathcal{Z}$ nor with the actual ideal functionalities in the simulation. It also does not interfere with the ideal functionalities simulated by $\mathcal{S}$ towards its internal copy of $\mathcal{A}$. The steps of $\mathcal{A}_1$ only affect its own copies of $\mathcal{Z}, \mathcal{A}$ inside this reduction and and $\mathcal{S}$ only *observes* queries to $\mathcal{F}_{\text{gRO1}}$ in its simulation. While this approach may seem counter-intuitive, similar techniques are used by Dodis *et al.* [20] and shown to be compatible with the Global UC framework.

$\mathcal{A}_1$ waits for pk_0, s' from $\mathcal{A}$ and, if $s \neq s'$, rewinds $\mathcal{A}$ to the the previous step, randomly picks a different query (sid, s) from the set of $\mathcal{A}$'s queries to $\mathcal{F}_{\text{gRO1}}$ and answers it with (sid, q), repeating the same procedure after receiving a new pk_0, s' from $\mathcal{A}$. Notice that $\mathcal{A}_1$ (but *not* $\mathcal{S}$) rewinds *copies* of both the environment $\mathcal{Z}$ and the adversary $\mathcal{A}$ only as a step of this reduction where these copies are used in a black-box way to break Property 1. $\mathcal{A}_1$ does not interact with the actual environment $\mathcal{Z}$ or $\mathcal{S}$'s copy of $\mathcal{A}$ but only with its own copies of $\mathcal{A}$ and $\mathcal{Z}$ towards which it emulates all the ideal functionalities and parties. The steps of $\mathcal{A}_1$ only affect its own copies of $\mathcal{Z}, \mathcal{A}$ inside this reduction and no rewinding is done by $\mathcal{S}$ in the simulation of π_{SFOT}. While this approach may seem counter-intuitive, similar techniques are used by Dodis *et al.* [20] and shown to be compatible with the Global UC framework.

If $s = s'$, $\mathcal{A}_1$ computes pk_1 and sends $\mathsf{pk}_0, \mathsf{pk}_1$ to the challenger as public keys $\mathsf{pk}_0, \mathsf{pk}_1$ of the game. Upon receiving ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ from the challenger in the game of Property 1, $\mathcal{A}_1$ samples $\mathsf{ch} \xleftarrow{\$} \{0,1\}^{\kappa}$ and sends $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}'_0, \mathsf{ct}'_1)$ to $\mathcal{A}$. Upon receiving a query $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ to $\mathcal{F}_{\text{gRO3}}$ or $\mathcal{F}_{\text{gRO2}}$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_{i,j}; \mathsf{r}_{i,j})$, $\mathcal{A}_1$ adds $(\mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ to an initially empty list L_i and answers the query to $\mathcal{F}_{\text{gRO3}}$ as $\mathsf{p}'_{i,j} \xleftarrow{\$} \{0,1\}^{\kappa}$ and the query to $\mathcal{F}_{\text{gRO2}}$ as $\widetilde{\mathsf{p}_{i,j}} \xleftarrow{\$} \{0,1\}^{\lambda}$. Upon receiving a query $(\mathsf{sid}, \mathsf{p}'_{i,j})$ from $\mathcal{A}$ to $\mathcal{F}_{\text{gRO3}}$, if p''_{1-i} is not defined, $\mathcal{A}_1$ answers with $\mathsf{p}''_{i,j} \xleftarrow{\$} \{0,1\}^{\kappa}$. Otherwise, it answers with $\mathsf{p}''_{1-i,j}$ such that $\mathsf{ch} = \mathsf{p}''_{0,j} \oplus \mathsf{p}''_{1,j}$. Upon receiving $(\mathsf{sid}, \mathsf{chr})$ from $\mathcal{A}$, $\mathcal{A}_1$ checks that $\mathsf{chr} = \mathsf{p}''_{0,j}$ or $\mathsf{ch} \oplus \mathsf{p}''_{1,j}$ (for one of the values $\mathsf{p}''_{i,j}$ given as answer from $\mathcal{F}_{\text{gRO3}}$), failing otherwise. $\mathcal{A}_1$ samples $\mathsf{m}_0, \mathsf{m}_1 \xleftarrow{\$} \{0,1\}^{\lambda}$ and, for $i \in \{0,1\}$, queries $\mathcal{F}_{\text{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ obtaining $\widetilde{\mathsf{p}_{i,j}}$ (simulating this answer accoding to the procedure described before) and computes $\widetilde{\mathsf{m}_i} = \widetilde{\mathsf{p}_{i,j}} \oplus \mathsf{m}_i$. $\mathcal{A}_1$ sends $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_{0,j}, \mathsf{p}'_{1,j})$ to $\mathcal{A}$. Finally, when $\mathcal{A}$ terminates, $\mathcal{A}_1$ chooses a random $\mathsf{p}_0 \in L_0$ and $\mathsf{p}_1 \in L_1$ and sends $(\mathsf{p}_0, \mathsf{p}_1)$ to the challenger in the game of Property 1.

Notice that the queries to $\mathcal{F}_{\mathrm{gRO3}}$ will appear consistent with $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}'_0, \mathsf{ct}'_1)$. Moreover, notice that, for each $i \in \{0,1\}$, an adversary $\mathcal{A}$ that obtains any information about m_i with probability p must first recover $(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ from ct_i also with probability p. Due to Property 4, for any arbitrary public key pk_i, $\mathcal{A}$ can only both obtain $(\mathsf{p}_i, \mathsf{r}_i)$ from ct_i such that $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ and generate $n-1$ alternative message and randomness pairs $(\mathsf{p}_{i,1}, \mathsf{r}_{i,1}), \ldots, (\mathsf{p}_{i,n-1}, \mathsf{r}_{i,n-1})$ different from $(\mathsf{p}_i, \mathsf{r}_i)$ such that $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ for $j = 1, \ldots, n-1$ with probability $\leq \frac{1}{n} + \mathsf{negl}(\kappa)$. Notice that an adversary that can generate $n_i - 1$ such alternative message and randomness pairs for a ciphertext generated under public key pk_i can only recover pair $\mathsf{p}_i, \mathsf{r}_i$ necessary for obtaining m_i with probability $\frac{1}{n}$. Hence, an $\mathcal{A}$ who is able to generate $n_0 - 1$ (resp. $n_1 - 1$) such alternative message and randomness pairs for a ciphertext generated under public key pk_0 (resp. pk_1) can only both query $\mathcal{F}_{\mathrm{gRO3}}$ with $(\mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ and later manage to query $\mathcal{F}_{\mathrm{gRO2}}$ with $(\mathsf{pk}_{1-c}, \mathsf{p}_{1-c}, \mathsf{r}_{1-c})$ with probability $\leq \frac{1}{n_0}\frac{1}{n_1} + \mathsf{negl}(\kappa)$. Without loss of generality, we assume that $n = n_0 = n_1$. Notice that for an $\mathcal{A}$ that does this, L_0 and L_1 must contain $(\mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{p}_1, \mathsf{r}_1)$ such that $\mathsf{p}_0, \mathsf{p}_1$ win the game against the challenger of Property 1, since it must extract the exact pairs $(\mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{p}_1, \mathsf{r}_1)$ used in generating ct_0 and ct_1. Since $\mathcal{A}_1$ samples a random pair $(\mathsf{p}_i, \mathsf{r}_i)$ from the list L_i of all pairs $(\mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ that result in ct_i under pk_i, if L_i has n elements, it selects both the correct value p_0 and value p_1 with probability $\frac{1}{n^2}$. However, if $\mathcal{A}$ is also able to generate $n-1$ alternative pairs $(\mathsf{p}_{i,j}, \mathsf{r}_{i,j})$ that result in ct_i under pk_i, it is also only able to recover $(\mathsf{p}_i, \mathsf{r}_i)$ from ct_i with probability $\frac{1}{n}$ and, consequently, only able to recover both $(\mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{p}_1, \mathsf{r}_1)$ with probability $\frac{1}{n^2}$. Hence, $\mathcal{A}_1$ wins the game of Property 1 with the same probability that $\mathcal{Z}$ in the pair of $\mathcal{A}$ and $\mathcal{Z}$ distinguishes the ideal execution with $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ and $\mathcal{S}$ from a real execution with $\mathcal{A}$ and π_{SFOT}. Notice that $\mathcal{A}_1$ needs to rewind its own *copies* of the environment $\mathcal{Z}$ and the adversary $\mathcal{A}$ and program the random oracle it simulates towards these copies only as a step of the reduction where $\mathcal{Z}, \mathcal{A}$ are used to break Property 1. $\mathcal{A}_1$ does not interact with the actual environment $\mathcal{Z}$ nor with the actual (or simulated) ideal functionalities in the simulation. The steps of $\mathcal{A}_1$ only affect its own copies of $\mathcal{Z}, \mathcal{A}$ inside that specific reduction and no programming or rewinding is done by the simulator $\mathcal{S}$ in the simulation of π_{SFOT}. While this approach may seem counter-intuitive, similar techniques are used and shown to be compatible with the Global UC framework by Dodis *et al.* [20].

Simulator for the case only Alice *is corrupted.* In the case where only Alice is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ 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 receives queries from $\mathcal{A}$ to the instances of $\mathcal{F}_{\mathrm{gRO}}$ (*i.e.* $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$ and $\mathcal{F}_{\mathrm{gRO3}}$), which it forwards. When the query is answered, $\mathcal{S}$ forwards the answer to $\mathcal{A}$. The goal of the simulator is to extract the messages

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 $\mathcal{F}^{\lambda}_{\text{SFOT}}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Alice, proceeding as follows:

1. $\mathcal{S}$ forwards all messages between $\mathcal{A}$ and global random oracles $\mathcal{F}_{\text{gRO1}}$, $\mathcal{F}_{\text{gRO2}}$ and $\mathcal{F}_{\text{gRO3}}$. Moreover, it keeps up-to-date lists of adversarial and illegitimate queries (and associated answers) for each global random oracle.
2. $\mathcal{S}$ samples $c \xleftarrow{\$} \{0,1\}$ and follows the instructions of an honest Bob in Step 1 of Protocol π_{SFOT} to generate $(\mathsf{sid}, s, \mathsf{pk}_0)$: $\mathcal{S}$ generates a pair of keys $(\mathsf{pk}_c, \mathsf{sk}_c) \xleftarrow{\$} \mathsf{KG}(1^{\kappa})$, samples $s \xleftarrow{\$} \{0,1\}^{\kappa}$, sends (sid, s) to $\mathcal{F}_{\text{gRO1}}$, obtaining q as answer. Bob computes pk_{1-c} such that $\mathsf{pk}_0 \star \mathsf{pk}_1 = q$ and sends $(\mathsf{sid}, s, \mathsf{pk}_0)$ to $\mathcal{A}$.
3. Upon receiving $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1)$ from $\mathcal{A}$, $\mathcal{S}$ checks whether the challenge ch is valid or if $\mathcal{A}$ is trying to guess the choice bit by proceeding as follows:
   (a) For $i \in \{0,1\}$, $\mathcal{S}$ checks if there exist queries $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\text{gRO3}}$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$. If these checks fail for both $i \in \{0,1\}$, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \mathsf{force})$ to $\mathcal{F}^{\lambda}_{\text{SFOT}}$ (as an honest Bob would abort with overwhelming probability when checking the values $\mathsf{ch}, \mathsf{p}'_0, \mathsf{p}'_1$). If these checks succeed for both $i \in \{0,1\}$ and ch is computed correctly from $\mathsf{p}_0, \mathsf{p}_1$, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \perp)$ to $\mathcal{F}^{\lambda}_{\text{SFOT}}$ and goes to 3(c).
   (b) If there exists only one query $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$ to $\mathcal{F}_{\text{gRO3}}$ such that $\mathsf{ct}_{c'} = \mathsf{Enc}(\mathsf{pk}_{c'}, \mathsf{p}_{c'}; \mathsf{r}_{c'})$ or ch is invalid with respect to $\mathsf{p}_0, \mathsf{p}_1$ and $\mathcal{F}_{\text{gRO3}}$, $\mathcal{S}$ checks that there exists a query $(\mathsf{sid}, \mathsf{p}'_{c'})$ to $\mathcal{F}_{\text{gRO3}}$ with output $\mathsf{p}''_{c'}$ such that $\mathsf{p}'_{c'}$ was obtained as the output of a query $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$ to $\mathcal{F}_{\text{gRO3}}$ and that $\mathsf{p}''_{c'}$ satisfies $\mathsf{p}''_{c'} \oplus \mathsf{ch} = G'$, for $G' \in \{0,1\}^{\kappa}$ obtained as the output of a query (sid, G) to $\mathcal{F}_{\text{gRO3}}$. If such a query exists, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, c')$ to $\mathcal{F}^{\lambda}_{\text{SFOT}}$. Otherwise, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \mathsf{force})$.
   (c) If $(\mathsf{guess}, \mathsf{sid}, \perp)$ was sent to $\mathcal{F}^{\lambda}_{\text{SFOT}}$, $\mathcal{S}$ computes chr according to p_0 and $\mathcal{F}_{\text{gRO3}}$, sends $(\mathsf{sid}, \mathsf{chr})$ to $\mathcal{A}$ and goes to Step 4. Otherwise, $\mathcal{S}$ sets $c = c'$ (resp. $c = \tilde{c}$) if $\mathcal{F}^{\lambda}_{\text{SFOT}}$ answers with $(\mathsf{cheat} - \mathsf{undetected}, \mathsf{sid})$ (resp. $(\mathsf{cheat} - \mathsf{dectected}, \mathsf{sid}, \tilde{c})$). $\mathcal{S}$ computes chr according to $\mathsf{ch}, \mathsf{p}_c, c$ and $\mathcal{F}_{\text{gRO3}}$, and sends $(\mathsf{sid}, \mathsf{chr})$ to $\mathcal{A}$.
4. Upon receiving $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}'_0, \mathsf{p}'_1)$ from $\mathcal{A}$, if $(\mathsf{guess}, \mathsf{sid}, \perp)$ was sent to $\mathcal{F}^{\lambda}_{\text{SFOT}}$ or if $\mathcal{F}^{\lambda}_{\text{SFOT}}$ answered $(\mathsf{guess}, \mathsf{sid}, c)$ with $(\mathsf{cheat} - \mathsf{undetected}, \mathsf{sid})$, $\mathcal{S}$ runs the procedure of an honest Bob to check if $\mathsf{p}'_0, \mathsf{p}'_1$ are valid according to $\mathsf{ch}, \mathsf{p}_c, c$ (set as in step 3(c)) and $\mathcal{F}_{\text{gRO3}}$. If this check fails, $\mathcal{S}$ aborts as an honest Bob would. If $\mathcal{F}^{\lambda}_{\text{SFOT}}$ answered $(\mathsf{guess}, \mathsf{sid}, c)$ with $(\mathsf{cheat} - \mathsf{dectected}, \mathsf{sid}, \tilde{c})$, $\mathcal{S}$ allows the deliver of this message to Bob at this point. In the other cases, $\mathcal{S}$ sends $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ to $\mathcal{F}_{\text{gRO2}}$ obtaining $\widetilde{\mathsf{p}_c}$ as response and computing $\mathsf{m}_c = \widetilde{\mathsf{p}_c} \oplus \widetilde{\mathsf{m}_c}$. If $(\mathsf{guess}, \mathsf{sid}, \perp)$ was sent to $\mathcal{F}^{\lambda}_{\text{SFOT}}$, $\mathcal{S}$ sends $(\mathsf{sid}, \mathsf{pk}_{1-c}, \mathsf{p}_{1-c}, \mathsf{r}_{1-c})$ to $\mathcal{F}_{\text{gRO2}}$ obtaining $\widetilde{\mathsf{p}_{1-c}}$ as response and computing $\mathsf{m}_{1-c} = \widetilde{\mathsf{p}_{1-c}} \oplus \widetilde{\mathsf{m}_{1-c}}$; otherwise $\mathcal{S}$ sets $\mathsf{m}_{1-c} \xleftarrow{\$} \{0,1\}^{\lambda}$. Finally, $\mathcal{S}$ sends $(\mathsf{send}, \mathsf{sid}, \mathsf{m}_0, \mathsf{m}_1)$ to $\mathcal{F}^{\lambda}_{\text{SFOT}}$.
5. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

Fig. 7. Simulator $\mathcal{S}$ for the case where only Alice is corrupted.

of the corrupted sender that it needs to deliver to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$. Moreover, the simulator must extract the choice bit "guess" that the adversary might make and forward it to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$. Again, we will adopt an strategy where the simulator executes all the steps of an honest Bob exactly as in the real protocol but using a random choice bit, as well as observing the queries to $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$ and $\mathcal{F}_{\mathrm{gRO3}}$.

The first difference between the simulation performed by $\mathcal{S}$ and a real execution of π_{SFOT} is that $\mathcal{S}$ uses a random choice bit $c \overset{\$}{\leftarrow} \{0,1\}$ instead of the real one. This affects the messages sent in Step 2 and Step 3 of the simulation but we will show that the simulation with a random choice bit is indistinguishable from the real execution. In Step 2, pk_0 and pk_1 are computed exactly as in π_{SFOT} but a random pk_c is the valid public key. Nevertheless, the message $(\mathsf{sid}, s, \mathsf{pk}_0)$ (and the pk_1 it defines) are indistinguishable from those generated in a real execution because a valid pk_c is indistinguishable from an invalid pk_{1-c} by Property 2.

In the case of Step 3, we will show that we can extract $\mathcal{A}$'s guessed choice bit (if it attempts a guess), provide it to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ and generate a response chr consistent with a real execution. In Step 3(a), $\mathcal{S}$ extracts the pairs $(\mathsf{p}_i, \mathsf{r}_i)$ used to generate each ct_i from the list of queries $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\mathrm{gRO3}}$ using the fact that there must be exactly one pair $(\mathsf{p}_i, \mathsf{r}_i)$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, which is guaranteed by Property 3. Moreover, we leverage Property 5, which guarantees that an honest Bob would obtain the same $(\mathsf{p}_i, \mathsf{r}_i)$ from ct_i for $i = c$. First, if for both ct_i there is no query $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\mathrm{gRO3}}$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, then $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \mathsf{force})$ to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$, as an honest Bob would abort with overwhelming probability regardless of its choice bit c when he checks the consistency of the values p'_0 and p'_1 sent by $\mathcal{A}$ with p'_c that he computed locally and ch. Next, it checks that ch is consistent with $(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ used to generate ct_i, which means that $\mathcal{A}$ is not trying to guess the choice bit as a correctly computed ch consistent with $\mathsf{ct}_0, \mathsf{ct}_1$ has an answer chr that reveals no information about c. If everything is right, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \perp)$ to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$.

In case ch is not consistent with $\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i$ or for one ct_i there is no query $(\mathsf{sid}, \mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ to $\mathcal{F}_{\mathrm{gRO3}}$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$, $\mathcal{A}$ might be mounting a selective failure attack and $\mathcal{S}$ must extract the choice bit that $\mathcal{S}$ might be trying to guess. $\mathcal{A}$ can guess that an honest Bob's choice bit is i by providing ct_i with a consistent a challenge ch generated from $(\mathsf{pk}_i, \mathsf{p}_i, \mathsf{r}_i)$ and a random p'_{1-i} for which there is no query $(\mathsf{sid}, \mathsf{p}_{1-i})$ to $\mathcal{F}_{\mathrm{gRO3}}$ with output p'_{1-i}. In this scenario, if an honest Bob has choice bit i, it will be able to validate ct_i and ch with respect to $\mathsf{p}_i, \mathsf{p}'_{1-i}$, since it does not see any inconsistency between p'_{1-i} and the contents of ct_{1-i} (which he cannot recover). However, if an honest bob has choice bit $1-i$, he will be able to detect the inconsistency. Since $\mathcal{S}$ does not know the actual choice bit c, it tests $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ with respect to the $\mathsf{p}_0, \mathsf{r}_0, \mathsf{p}_1, \mathsf{r}_1$ extracted (or not) from the list of queries to $\mathcal{F}_{\mathrm{gRO3}}$ in order to check that $\mathcal{A}$ is trying to guess the choice bit is i using the strategy previously described. In case $\mathcal{S}$ detects that $\mathcal{A}$ is trying to guess that the choice bit is i, it sends $(\mathsf{guess}, \mathsf{sid}, i)$ to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$. On the other hand, if ch could not be generated by querying $\mathcal{F}_{\mathrm{gRO3}}$ with p_i and a random p'_{1-i}, an honest Bob would detect that $\mathcal{A}$ is cheating with all but negligible probability. Hence, $\mathcal{S}$ sends $(\mathsf{guess}, \mathsf{sid}, \mathsf{force})$ to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$. In any case, $\mathcal{S}$ uses the answer from $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ to set c and generate chr as an honest Bob would.

In Step 4, $\mathcal{S}$ checks the validity of ch as an honest Bob would. Namely, if $\mathcal{S}$ has not detected that $\mathcal{A}$ tried to guess the choice bit or if $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ answered with $(\mathsf{cheat} - \mathsf{undetected}, \mathsf{sid})$, $\mathcal{S}$ performs these checks as an honest Bob would using $\mathsf{p}'_0, \mathsf{p}'_1, \mathsf{p}_c, c$ (with c set in step 3(c)). If $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ answered with $(\mathsf{cheat} - \mathsf{detected}, \mathsf{sid})$, $\mathcal{S}$ aborts as an honest Bob would upon detecting a cheating Alice. Hence $\mathcal{S}$ validates ch as an honest Bob would with all but negligible probability.

It remains to show that $\mathcal{S}$ can extract the necessary messages in case the checks with ch succeed. If that happens, either (1) the message $(\mathsf{guess}, \mathsf{sid}, \perp)$ was sent to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ and $\mathcal{S}$ needs to extract both m_0 and m_1; or (2) the message $(\mathsf{guess}, \mathsf{sid}, c)$ was sent to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$ with the correct guess and $\mathcal{S}$ needs to extract m_c. In the first case, both $(\mathsf{p}_0, \mathsf{r}_0)$ and $(\mathsf{p}_1, \mathsf{r}_1)$ were already extracted from the list of queries to $\mathcal{F}_{\mathrm{gRO3}}$. Similarly, in the second case the values $(\mathsf{p}_c, \mathsf{r}_c)$ were extracted. Property 3 guarantees that there is a single pair $(\mathsf{p}_i, \mathsf{r}_i)$ such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{p}_i; \mathsf{r}_i)$ and Property 5 guarantees that this same pair would be recovered by an honest Bob with choice bit i. Hence, the simulator can use these values to query $\mathcal{F}_{\mathrm{gRO2}}$, obtain the necessary one-time pads and extract the correct messages to send to $\mathcal{F}^{\lambda}_{\mathrm{SFOT}}$.

C Security Analysis of Protocol π_{OT}

In this appendix we analyse the security of Protocol π_{OT} and present a full proof of Theorem 2, re-stated below.

Theorem 2 *Let* PKE *be a public-key encryption scheme that satisfies Properties 1, 2, 3, 4 and 5. When instantiated with* PKE*, Protocol* π_{OT} *UC-realizes the functionality* $\mathcal{F}^{\lambda,1}_{\mathrm{OT}}$ *against static malicious adversaries in the global random oracle model.*

Proof. In order to prove the security of π_{OT}, we will construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between interactions with an adversary $\mathcal{A}$ through π_{OT} in the real world and with $\mathcal{S}$ and $\mathcal{F}^{\lambda,1}_{\mathrm{OT}}$ in the ideal world. For the sake of clarity, we will describe the simulator $\mathcal{S}$ separately for 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 $\mathcal{Z}$. Notice that simulating the cases where both Bob and Alice are corrupted or honest is trivial. If both parties are corrupted, $\mathcal{S}$ simply runs $\mathcal{A}$ internally. In this case, $\mathcal{A}$ generates 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 $\mathcal{A}$. We analyze the cases where only Bob is corrupted and where only Alice is corrupted below.

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 $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Bob, proceeding as follows:

1. $\mathcal{S}$ forwards all messages between $\mathcal{A}$ and global random oracles $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{F}_{\mathrm{gRO3}}$ and $\mathcal{F}_{\mathrm{gRO4}}$. Moreover, it keeps up-to-date lists of adversarial and illegitimate queries (and associated answers) for each global random oracle.
2. Upon receiving $(\mathsf{sid}, s, \mathsf{pk}_0)$ from $\mathcal{A}$, $\mathcal{S}$ follows the instructions of an honest Alice in Step 2 of Protocol π_{OT} to generate and send $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1, \hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1)$ to $\mathcal{A}$. Notice that an honest Alice's inputs are not used in this step of Protocol π_{OT}, so $\mathcal{S}$'s message to $\mathcal{A}$ is distributed exactly as in a real execution of Protocol π_{OT}.
3. Whenever the first query $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$ from $\mathcal{A}$ to $\mathcal{F}_{\mathrm{gRO2}}$ or $\mathcal{F}_{\mathrm{gRO3}}$ where $c' \in \{0,1\}$ happens, $\mathcal{S}$ stores c' (*i.e.* the random choice bit used by $\mathcal{A}$, which will be used in Step 5 to determine the actual choice bit c).
4. Upon receiving $(\mathsf{sid}, \mathsf{chr})$ from $\mathcal{A}$, $\mathcal{S}$ checks that $\mathsf{chr} = \mathsf{p}_0''$, aborting otherwise (as an honest Alice would). If no query of the form $(\mathsf{sid}, \mathsf{pk}_c, \mathsf{p}_c, \mathsf{r}_c)$ has been recorded in the lists of $\mathcal{F}_{\mathrm{gRO3}}$ or $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{S}$ outputs fail and halts. Otherwise, $\mathcal{S}$ executes Step 4 of Protocol π_{OT} as an honest Alice to generate and send $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ to $\mathcal{A}$. Once again, notice that an honest Alice's inputs are not used in this step of Protocol π_{OT}, so that $\mathcal{S}$'s message to $\mathcal{A}$ is distributed exactly as in a real execution of Protocol π_{OT}.
5. Upon receiving (sid, d) from $\mathcal{A}$, $\mathcal{S}$ sends $(\mathsf{choose}, \mathsf{sid}, d \oplus c')$ to $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ (*i.e.* using choice bit $c = d \oplus c'$ where the random choice bit c' has been extracted in Step 3). Upon receiving $(\mathsf{output}, \mathsf{sid}, \mathsf{m}_c)$ from $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$, $\mathcal{S}$ samples a random message $\mathsf{m}_{1-c} \xleftarrow{\$} \{0,1\}^\lambda$ and uses $\mathsf{m}_c, \mathsf{m}_{1-c}$ as Alice's inputs to execute Step 6 of Protocol π_{OT} as an honest Alice, generating and sending $(\mathsf{sid}, \mathsf{m}_0', \mathsf{m}_1')$ to $\mathcal{A}$. Notice that the only deviation from Protocol π_{OT} is that m_{1-c} is sampled at random.
6. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

Fig. 8. Simulator $\mathcal{S}$ for the case where only Bob is corrupted.

Simulator for the case only Bob *is corrupted.* In the case where only Bob is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ 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 receives queries from $\mathcal{A}$ to the instances of $\mathcal{F}_{\mathrm{gRO}}$ (*i.e.* $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{F}_{\mathrm{gRO3}}$ and $\mathcal{F}_{\mathrm{gRO4}}$), which it forwards. When the query is answered, $\mathcal{S}$ forwards the answer to $\mathcal{A}$. The goal of the simulator is to extract the corrupted receiver's choice bit c in order to request the correct message from $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$, which is later transferred to the adversary. The simulator $\mathcal{S}$ is presented in Figure 8. Notice that, unless $\mathcal{S}$ outputs fail, it executes all the steps of an honest Alice in the real protocol with the sole difference that it uses a random message

$\mathsf{m}_{1-c} \xleftarrow{\$} \{0,1\}^\lambda$ instead of the real message. We will show that $\mathcal{S}$ only outputs fail with negligible probability and that simulation with a uniformly random m_{1-c} is indistinguishable from a real execution.

The fact that $\mathcal{S}$ only outputs fail with negligible probability has been established in the proof of Theorem 1. First, notice that $\mathcal{S}$'s challenge consisting of $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ is generated exactly as in Protocol π_{SFOT}. In the proof of Theorem 1 it is shown that $\mathcal{A}$ can only provide a valid response $(\mathsf{sid}, \mathsf{chr})$ to this challenge if it first queries $\mathcal{F}_{\mathrm{gRO2}}$ or $\mathcal{F}_{\mathrm{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$, except with negligible probability. Moreover, it is shown that, if $\mathcal{A}$'s first such query to $\mathcal{F}_{\mathrm{gRO2}}$ or $\mathcal{F}_{\mathrm{gRO3}}$ is $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$, then it can only decrypt ciphertext $\mathsf{ct}_{c'}$ (*i.e.* it only has secret key $\mathsf{sk}_{c'}$) but not ciphertext $\mathsf{ct}_{1-c'}$.

In order to show that our simulation with a uniformly random m_{1-c} is indistinguishable from a real execution, we first argue that $\mathcal{A}$ cannot decrypt $\hat{\mathsf{ct}}_{1-c'}$. Notice that $\hat{\mathsf{ct}}_{1-c'}$ (resp. $\hat{\mathsf{ct}}_{c'}$) is generated under public key $\mathsf{pk}_{1-c'}$ (resp. $\mathsf{pk}_{c'}$), which is the same public key used to generate $\mathsf{ct}_{1-c'}$ (resp. $\mathsf{ct}_{c'}$). Hence, we can repeat the argument from the proof of Theorem 1 to show that $\mathcal{A}$ cannot decrypt $\hat{\mathsf{ct}}_{1-c'}$ (and can only decrypt $\hat{\mathsf{ct}}_{c'}$) if it has first queried $\mathcal{F}_{\mathrm{gRO2}}$ or $\mathcal{F}_{\mathrm{gRO3}}$ with $(\mathsf{sid}, \mathsf{pk}_{c'}, \mathsf{p}_{c'}, \mathsf{r}_{c'})$. Since $\mathcal{A}$ cannot decrypt $\hat{\mathsf{ct}}_{1-c'}$, it cannot recover $\widetilde{\mathsf{p}_{1-c'}}$, which it needs in order to compute $\hat{\mathsf{m}}_{1-c'} \leftarrow \widetilde{\mathsf{m}_{1-c'}} \oplus \widetilde{\mathsf{p}_{1-c'}}$ and subsequently $\mathsf{m}_{1-c} = \mathsf{m}'_{1-c} \oplus \hat{\mathsf{m}}_{1-c'}$. Hence m'_{1-c} appears uniformly distributed towards $\mathcal{A}$ regardless of m_{1-c}, which $\mathcal{S}$ samples uniformly at random (as opposed to using Alice's actual input as in Protocol π_{OT}). On the other hand, $\mathcal{S}$ recovers the correct message m_c from $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ using $\mathcal{A}$'s message $(\mathsf{d}, \mathsf{sid},)$ and c', resulting in $\mathcal{A}$ obtaining the same message m_c as it would in the real protocol execution.

Simulator for the case only Alice *is corrupted.* In the case where only Alice is corrupted, the simulator $\mathcal{S}$ interacts with $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ 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 receives queries from $\mathcal{A}$ to the instances of $\mathcal{F}_{\mathrm{gRO}}$ (*i.e.* $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{F}_{\mathrm{gRO3}}$ and $\mathcal{F}_{\mathrm{gRO4}}$), which it forwards. When the query is answered, $\mathcal{S}$ forwards the answer to $\mathcal{A}$. The goal of the simulator is to extract the messages of the corrupted sender that it needs to deliver to $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$. Again, we will adopt an strategy where the simulator executes all the steps of an honest Bob exactly as in the real protocol but using a random choice bit, as well as observing the queries to $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{F}_{\mathrm{gRO3}}$ and $\mathcal{F}_{\mathrm{gRO4}}$. We argue that such a simulation with a random choice bit is indistinguishable from a real execution, since the (random) choice bit does not leak and both of $\mathcal{A}$'s messages can be extracted if $\mathcal{S}$ does not abort, which it only does if an honest Bob would abort.

First, as in Protocol π_{SFOT}, we remark that message $(\mathsf{sid}, s, \mathsf{pk}_0)$ does not leak information about the random choice bit c' used in the first step of Protocol π_{OT} due to Property 2. The only way $\mathcal{A}$ can learn c' is by mounting a selective failure attack where it generates the challenge consisting of $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ maliciously in such a way that it can guess c' and confirm its guess by observing $\mathcal{S}$'s response chr to the challenge. Notice that this is the selective failure attack to which Protocol π_{SFOT} is vulnerable. However, in Protocol π_{OT}, $\mathcal{A}$ must reveal the messages and randomness $\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1$ used to generate the challenge

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

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 $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ and an internal copy $\mathcal{A}$ of the adversary that corrupts only Alice, proceeding as follows:

1. $\mathcal{S}$ forwards all messages between $\mathcal{A}$ and global random oracles $\mathcal{F}_{\mathrm{gRO1}}$, $\mathcal{F}_{\mathrm{gRO2}}$, $\mathcal{F}_{\mathrm{gRO3}}$ and $\mathcal{F}_{\mathrm{gRO4}}$. Moreover, it keeps up-to-date lists of adversarial and illegitimate queries (and associated answers) for each global random oracle.
2. $\mathcal{S}$ follows the instructions of an honest Bob in Step 1 of Protocol π_{OT} to compute $(\mathsf{sid}, s, \mathsf{pk}_0)$ and send it to $\mathcal{A}$. Notice that the choice bit of an honest Bob is not used at this step of Protocol π_{OT}, so $\mathcal{S}$'s message is distributed exactly as in the real execution.
3. Upon receiving $(\mathsf{sid}, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1, \hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1)$ from $\mathcal{A}$, $\mathcal{S}$ follows the instructions of an honest Bob in Step 3 of Protocol π_{OT} to compute $(\mathsf{sid}, \mathsf{chr})$ and send it to Alice. Notice that the choice bit of an honest Bob is not used at this step of Protocol π_{OT}, so $\mathcal{S}$'s message is distributed exactly as in the real execution.
4. Upon receiving $(\mathsf{sid}, \widetilde{\mathsf{m}_0}, \widetilde{\mathsf{m}_1}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ from $\mathcal{A}$, $\mathcal{S}$ samples a random choice bit $c \overset{\$}{\leftarrow} \{0,1\}$ and follows the steps of an honest Bob in Step 5 of Protocol π_{OT} to compute (sid, d) and send it to Alice (aborting if an honest Bob would have aborted).
5. Upon receiving $(\mathsf{sid}, \mathsf{m}'_0, \mathsf{m}'_1)$ from $\mathcal{A}$, $\mathcal{S}$ proceeds as follows:
   (a) For $i \in \{0,1\}$, check that there exists a query $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i)$ to $\mathcal{F}_{\mathrm{gRO4}}$ with answer $\hat{\mathsf{r}}_i$ such that $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_i; \hat{\mathsf{r}}_i)$ (checking that ciphertexts $\hat{\mathsf{ct}}_i$ were correctly generated from values $\hat{\mathsf{p}}_i$ using randomness $\hat{\mathsf{r}}_i$ obtained by querying $\mathcal{F}_{\mathrm{gRO4}}$ with $\hat{\mathsf{p}}_i$).
   (b) If the previous check fails for $i \in \{0,1\}$, set $\widetilde{\mathsf{p}_i} \overset{\$}{\leftarrow} \{0,1\}^\lambda$ (since in this case an honest Bob with choice bit $c = i$ would output a random message). For $i \in \{0,1\}$, if the check succeed obtaining values $\hat{\mathsf{p}}_i, \hat{\mathsf{r}}_i$, send query $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i, \hat{\mathsf{r}}_i)$ to $\mathcal{F}_{\mathrm{gRO2}}$ obtaining answer $\widetilde{\mathsf{p}_i}$.
   (c) For $i \in \{0,1\}$, compute $\hat{\mathsf{m}}_i = \widetilde{\mathsf{p}_i} \oplus \widetilde{\mathsf{m}_i}$ and $\mathsf{m}_i = \mathsf{m}'_i \oplus \hat{\mathsf{m}}_i$ as an honet Bob would.
   (d) Send $(\mathsf{send}, \mathsf{sid}, \mathsf{m}_0, \mathsf{m}_1)$ to $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$.
6. When $\mathcal{A}$ halts, $\mathcal{S}$ also halts and outputs whatever $\mathcal{A}$ outputs.

Fig. 9. Simulator $\mathcal{S}$ for the case where only Alice is corrupted.

$\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ so that $\mathcal{S}$ (following the steps of an honest Bob) can check that the challenge was constructed correctly and that consequently its response chr does not leak any information about c'. Due to Property 3, $\mathcal{A}$ cannot generate alternative messages and randomness $(\mathsf{p}'_0, \mathsf{p}'_1, \mathsf{r}'_0, \mathsf{r}'_1) \neq (\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ that result in the same challenge $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ such that $(\mathsf{p}'_0, \mathsf{p}'_1, \mathsf{r}'_0, \mathsf{r}'_1)$ passes the check performed by $\mathcal{S}$ (*i.e.* the check of an honest Bob) but in fact $(\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1)$ is maliciously constructed and used to generate $\mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ in such a way that chr reveals c'. Hence, if $\mathcal{S}$ does not abort in Step 4 (due to the check on $\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ failing), $\mathcal{A}$ can only learn anything about c' if it breaks Property 2 and Property 3.

In Step 4, $\mathcal{S}$ only aborts if an honest Bob would have aborted due to the check on $\mathsf{p}_0, \mathsf{p}_1, \mathsf{r}_0, \mathsf{r}_1, \mathsf{ch}, \mathsf{ct}_0, \mathsf{ct}_1$ failing, so that the abort would be indistinguishable of an execution of Protocol π_{OT} with an honest Bob. If this check succeeds and $\mathcal{S}$ does not abort, we have established that $\mathcal{A}$ would not have learnt anything about c'. Hence, the message (sid, d) where $d = c \oplus c'$ and $c \xleftarrow{\$} \{0,1\}$ sent to $\mathcal{A}$ by $\mathcal{S}$ is indistinguishable from the message sent by an honest Bob, since $\mathcal{A}$ does not know c' (which acts as a one-time pad key in d) and consequently cannot distinguish d generated from a random c (as done by $\mathcal{S}$) from the d generated by an honest Bob.

Finally, having established that a simulation with a random c is indistinguishable from an real execution with an honest Bob, we show that $\mathcal{S}$ correctly extracts $\mathcal{A}$'s messages $\mathsf{m}_0, \mathsf{m}_1$. Notice that both pairs of ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ and $\hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1$ are generated with the same pair of public keys $\mathsf{pk}_0, \mathsf{pk}_1$. Due to the underlying PKE being committing (Property 3), for $i \in \{0,1\}$, there exists a unique triple $(\mathsf{pk}_i, \hat{\mathsf{p}}_i, \hat{\mathsf{r}}_i)$ such that $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_i; \hat{\mathsf{r}}_i)$ unless $\mathcal{A}$ breaks Property 3, which only happens with negligible probability. Hence, if there is a query $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i)$ to $\mathcal{F}_{\mathrm{gRO4}}$ with answer $\hat{\mathsf{r}}_i$ such that $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_i; \hat{\mathsf{r}}_i)$, there is a unique value $\widetilde{\mathsf{p}_i}$ that an honest Bob with $c' = i$ obtains by successfully decrypting $\hat{\mathsf{ct}}_i$ and querying $\mathcal{F}_{\mathrm{gRO2}}$ with $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i, \hat{\mathsf{r}}_i)$. In this case, $\mathcal{S}$ obtains exactly the same $\widetilde{\mathsf{p}_i}$ that an honest Bob with $c' = i$ would obtain. On the other hand, if there is no query $(\mathsf{sid}, \mathsf{pk}_i, \hat{\mathsf{p}}_i)$ to $\mathcal{F}_{\mathrm{gRO4}}$ with answer $\hat{\mathsf{r}}_i$ such that $\hat{\mathsf{ct}}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\mathsf{p}}_i; \hat{\mathsf{r}}_i)$, decryption would fail and an honest Bob with $c' = i$ would obtain $\perp$ when computing $(\hat{\mathsf{p}}_{c'}, \hat{\mathsf{r}}_{c'}) \leftarrow \mathsf{Dec}(\mathsf{sk}_{c'}, \hat{\mathsf{ct}}_{c'})$. In this case, $\mathcal{S}$ obtains a random value $\widetilde{\mathsf{p}_i}$, which is distributed exactly as the value an honest Bob with $c' = i$ would obtain in the protocol. Once an honest Bob (and $\mathcal{S}$) obtain $\widetilde{\mathsf{p}_0}, \widetilde{\mathsf{p}_1}$, the rest of the computation is deterministic since it only involves computing $\hat{\mathsf{m}}_i = \widetilde{\mathsf{p}_i} \oplus \widetilde{\mathsf{m}_i}$ and $\mathsf{m}_i = \mathsf{m}'_i \oplus \hat{\mathsf{m}}_i$. Hence, $\mathcal{S}$ obtains $\mathsf{m}_0, \mathsf{m}_1$ and sends $(\mathsf{send}, \mathsf{sid}, \mathsf{m}_0, \mathsf{m}_1)$ to $\mathcal{F}_{\mathrm{OT}}^{\lambda,1}$ in a way indistinguishable from a real world execution with $\mathcal{A}$. This completes the proof that an ideal world simulation with $\mathcal{S}$ is indistinguishable from a real world execution of Protocol π_{OT} with $\mathcal{A}$.

D Instantiating the Protocol with the CDH Assumption

We will instantiate our scheme by showing that the ElGamal cryptosystem is OW-CPA secure and has Properties 1, 2, 3, 4 and 5 under the Computational Diffie-Hellman (CDH) assumption in the Global Random Oracle model. First we will recall the CDH assumption:

Assumption 1 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 [27] 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 [27], leaving us to prove that it has Properties 1, 2, 3, 4 and 5. Property 2 follows trivially from the fact that pk is chosen uniformly over all elements of $\mathbb{G}$.

Observation 1 The ElGamal cryptosystem described above satisfies Property 1 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 1. 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.

Alternatively, we can construct an algorithm $\mathcal{B}$ solving the CDH problem from an adversary $\mathcal{A}$ breaking Property 1. $\mathcal{B}$ plays the game of Property 1 acting as the challenger towards $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Given a CDH instance (g, g^x, g^y), B sets $q = g^x$ and gives q to $\mathcal{A}_1$, who outputs state st and public keys $\mathsf{pk}_0 = g_0^{\mathsf{sk}}, \mathsf{pk}_1 = g_1^{\mathsf{sk}}$ such that $\mathsf{pk}_0 \cdot \mathsf{pk}_1 = q = g^x$ (*i.e.* $\mathsf{sk}_0 + \mathsf{sk}_1 = x$). $\mathcal{B}$ computers $\mathsf{ct}_0 = (g^y, h_0) and ct_1 = (g^{yr}, h_1)$, where $r \xleftarrow{\$} \mathbb{Z}_q$ and $h_0, h_1 \xleftarrow{\$} \mathbb{G}$, and gives $\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{st}$ to $\mathcal{A}_2$. If $\mathcal{A}$ breaks Property 1, it outputs two correct messages $\mathsf{m}_0, \mathsf{m}_1$. $\mathcal{B}$ computes $g^{xy} = \frac{h_0}{\mathsf{m}_0} \cdot \frac{h_1}{\mathsf{m}_1}^{\frac{1}{r}} = g^{(x - \mathsf{sk}_1)y} \cdot g^{\mathsf{sk}_1 y}$.

Properties 3 and 4 hold for the ElGamal cryptosystem as the ciphertext is a one-to-one function of the plaintext and randomness. Property 5 can be achieved using the encrypt-with-hash technique as discussed below.

D.1 Obtaining Witness-Recovering Decryption

Not all OW-CPA PKE schemes with the other properties we require immediately have Property 5. Nevertheless, for encryption scheme that do not enjoy witness-recovering decryption, we can obtain it with the following technique. By using the encrypt-with-hash technique [2] in the global random oracle model, it is possible to obtain a secure public-key encryption scheme with witness-recovering and that satisfies Property 1 given any secure public-key encryption that satisfies Property 1. Let $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a secure public-key encryption that satisfies Property 1 and let $\mathcal{F}_{\text{gRO}}$ be a global random oracle with outputs in $\mathcal{R}$. Now define the modified cryptosystem $\mathsf{PKE}' = (\mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$ with: (1) KG' that is the same as KG except that it includes the public key pk into the secret key sk; (2) an encryption procedure Enc' that given a public key pk and a message m, first queries $\mathcal{F}_{\text{gRO}}$ with input $(\mathsf{sid}, \mathsf{pk}, \mathsf{m})$ to get an output r and outputs ct for $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$; (3) a decryption procedure Dec' that given a secret key sk and a ciphertext ct first computes $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$, outputting $\perp$ if Dec outputs $\perp$. Then it queries $\mathcal{F}_{\text{gRO}}$ with input $(\mathsf{sid}, \mathsf{m})$ to get an output r and checks if $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r}) = \mathsf{ct}$, outputting $\perp$ if they are not equal; and (m, r) otherwise.

Note that since $\mathcal{M}$ is exponentially large in the security parameter (since otherwise an adversary $\mathcal{A}$ against PKE could trivially break its Property 1) and in the game defining Property 1 the challenge messages are independent and chosen uniformly at random in $\mathcal{M}$, the probability that the challenge messages $\mathsf{m}_1, \mathsf{m}_2$ are equal (resulting in the same ciphertext) is negligible. Basically, the only extra advantage that the adversary has when attacking PKE' is that it can test whether a challenge ciphertext encrypts a given message m by encrypting it and checking whether the resulting ciphertext matches the challenge ciphertext. However, the adversary can only test if a challenge ciphertext contains a given message m by performing a re-encryption test if he queries $\mathcal{F}_{\text{gRO}}$ with input $(\mathsf{sid}, \mathsf{m})$. Given that there is an exponential number of messages in $\mathcal{M}$ and that the adversary can only perform a polynomial number of queries to $\mathcal{F}_{\text{gRO}}$, it is trivial to get an adversary $\mathcal{A}$ that breaks Property 1 of PKE from an adversary $\mathcal{A}'$ that breaks Property 1 of PKE'.