Round-Optimal Secure Two-Party Computation from Trapdoor Permutations

Michele Ciampi
DIEM
Università di Salerno
ITALY
mciampi@unisa.it

Rafail Ostrovsky
UCLA
Los Angeles
rafail@cs.ucla.edu

Luisa Siniscalchi
DIEM
Università di Salerno
ITALY
lsiniscalchi@unisa.it

Ivan Visconti
DIEM
Università di Salerno
ITALY
visconti@unisa.it

Abstract

In this work we continue the study on the round complexity of secure two-party computation with black-box simulation.

Katz and Ostrovsky in CRYPTO 2004 showed a 5 (optimal) round construction assuming trapdoor permutations for the general case where both players receive the output. They also proved that their result is round optimal. This lower bound has been recently revisited by Garg et al. in Eurocrypt 2016 where a 4 (optimal) round protocol is showed assuming a simultaneous message exchange channel. Unfortunately there is no instantiation of the protocol of Garg et al. under standard polynomial-time hardness assumptions.

In this work we close the above gap by showing a 4 (optimal) round construction for secure two-party computation in the simultaneous message channel model with black-box simulation, assuming trapdoor permutations against polynomial-time adversaries.

Our construction for secure two-party computation relies on a special 4-round protocol for oblivious transfer that nicely composes with other protocols in parallel. We define and construct such special oblivious transfer protocol from trapdoor permutations. This building block is clearly interesting on its own. Our construction also makes use of a recent advance on non-malleability: a delayed-input 4-round non-malleable zero knowledge argument.

Contents

1 Introduction

Obtaining round-optimal secure computation [Yao82, GMW87] has been a long standing open problem. For the two-party case the work of Katz and Ostrovsky [KO04] demonstrated that 5 rounds are both necessary and sufficient, with black-box simulation, when both parties need to obtain the output. Their construction relies on the use of trapdoor permutations[1]. A more recent work of Ostrovsky et al. [ORS15] showed that a black-box use of trapdoor permutations is sufficient for obtaining the above round-optimal construction.

A very recent work of Garg et al. [GMPP16b] revisited the lower bound of [KO04] when the communication channel allows both players to send messages in the same round, a setting that has been widely used when studying the round complexity of multi-party computation. Focusing on the simultaneous message exchange model, Garg et al. showed that 4 rounds are necessary to build a secure two-party computation (2PC) protocol for every functionality with black-box simulation. In the same work they also designed a 4-round secure 2PC protocol for every functionality. However their construction compared to the one of [KO04] relies on much stronger complexity assumptions. Indeed the security of their protocol crucially relies on the existence of a 3-round 3-robust [GMPP16a, Pol16] parallel non-malleable commitment scheme. According to [GMPP16a, Pol16] such commitment scheme can be constructed either through non-falsifiable assumptions (i.e., using the construction of [PPV08]) or through sub-exponentially-strong assumptions (i.e., using the construction of [COSV16]). A recent work of Ananth et al. [ACJ17] studies the multi-party case in the simultaneous message exchange channel. More precisely the authors of [ACJ17] provide a 5-round protocol to securely compute every functionality for the multi-party case under the Decisional Diffie-Hellman (DDH) assumption and a 4-round protocol assuming one-way permutations and sub-exponentially secure DDH. The above gap in the state of affairs leaves open the following interesting open question:

Open Question: *is there a 4-round construction for secure 2PC for any functionality in the simultaneous message exchange model assuming (standard) trapdoor permutations?*

1.1 Our Contribution

In this work we solve the above open question. Moreover our construction only requires black-box simulation and is therefore round optimal. We now describe our approach.

As discussed before, the construction of [GMPP16b] needs a 3-round 3-robust parallel non-malleable commitment, and constructing this primitive from standard polynomial-time assumptions is still an open problem. We circumvent the use of this primitive through a different approach. As done in [GMPP16b], we start considering the 4-round 2PC protocol of [KO04] (KO protocol) that works only for those functionalities where only one player receives the output (we recall that the KO protocols do not assume the existence of a simultaneous message exchange channel). Then as in [GMPP16b] we consider two simultaneous executions of the KO protocol in order to make both the parties able to obtain the output *assuming the existence of a simultaneous message exchange channel*. We describe now the KO protocol and then we explain how we manage to avoid 3-round 3-robust parallel non-malleable commitments.

The 4-round KO protocol. Following Fig. 1, at a very high level the KO protocol between the players P_1 and P_2, where only P_1 gets the output, works as follows. Let f be the function that P_1 and

[1]The actual assumption is *enhanced* trapdoor permutations, but for simplicity in this paper we will omit the word *enhanced* assuming it implicitly.

P_2 want to compute. In the second round P_2 generates, using his input, a Yao's garbled circuit C for the function f with the associated labels L. Then P_2 commits to C using a commitment scheme that is binding if P_2 runs the honest committer procedure. This commitment scheme however admits also an indistinguishable equivocal commitment procedure that allows later to open the equivocal commitment as any message. Let $\mathtt{com}_0$ be such commitment. In addition P_2 commits to L using a statistically binding commitment scheme. Let $\mathtt{com}_1$ be such commitment. In the last round P_2 sends the opening of the equivocal commitment to the message C. Furthermore, using L as input, P_2 in the 2nd and in the 4th round runs as a sender of a specific 4-round oblivious transfer protocol KOOT that is secure against a malicious receiver and secure against a semi-honest sender. Finally, in parallel with KOOT, P_2 computes a specific delayed-input zero-knowledge argument of knowledge (ZKAoK) to prove that the labels L committed in $\mathtt{com}_1$ correspond to the ones used in KOOT, and that $\mathtt{com}_0$ is binding since it has been been computed running the honest committer on input some randomness and the message C. P_1 plays as a receiver of KOOT in order to obtain the labels associated to his input and computes the output of the two-party computation by running C on input the received labels. Moreover P_1 acts as a verifier for the ZKAoK where P_2 acts as a prover.

The 4-round protocol of Garg et al. In order to allow both parties to get the output in 4 rounds using a simultaneous message exchange channel, [GMPP16b] first considers two simultaneous execution of the KO protocol (Fig. 2). Such natural approach yields to the following two problems (as stated in [GMPP16b]): 1) nothing prevents an adversary from using two different inputs in the two executions of the KO protocol; 2) an adversary could adapt his input based on the input of the other party, for instance the adversary could simply forward the messages that he receives from the honest party. To address the first problem the authors of [GMPP16b] add another statement to the ZKAoK where the player P_j (with $j = 1, 2$) proves that both executions of the KO protocol use the same input. The second problem is solved in [GMPP16b] by using a 3-round 3-robust non-malleable commitment to construct KOOT and the ZKAoK in such a way that the input used by the honest party in KOOT cannot be mauled by the malicious party. The 3-robustness is required to avoid rewinding issues in the security proof. Indeed, in parallel with the 3-round 3-robust non-malleable commitment a WIPoK is executed in KOOT. At some point the security proof of [GMPP16b] needs to rely on the witness-indistinguishability property of the WIPoK while the simulator of the ZKAoK is run. The simulator for the ZKAoK rewinds the adversary from the third to the second round, therefore rewinding also the challenger of the WIPoK of the reduction. To solve this problem [GMPP16b, Pol16] rely on the stronger security of a 3-round 3-robust parallel non-malleable commitment scheme. Unfortunately, constructing this tool with standard polynomial-time assumptions is still an open question.

Our 4-round protocol. In our approach (that is summarized in Fig. 3), in order to solve problems 1 and 2 listed above using standard polynomial-time assumption (trapdoor permutations), we replace the ZKAoK and KOOT (that uses the 3-round 3-robust parallel commitment scheme) with the following two tools. 1) A 4-round delayed-input non-malleable zero-knowledge (NMZK) argument of knowledge (AoK) NMZK from one-way functions (OWFs) recently constructed in [COSV17a] (the theorem proved by NMZK is roughly the same as the theorem proved by ZKAoK of [GMPP16b]). 2) A new special OT protocol $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ that is *one-sided* simulatable [ORS15]. In this security notion for OT it is not required the existence of a simulator against a malicious sender, but only that a malicious sender cannot distinguish whether the honest receiver uses his real input or a fixed input (e.g., a string of 0s). Moreover some security against a malicious sender still holds even if the adversary can perform a mild form of "rewinds" against the receiver, and the security against a malicious receiver holds even when an interactive primitive (like a WIPoK) is run in parallel (more details about the security provided by $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ will be provided later).

Our security proof. In our security proof we exploit immediately the major differences with [GMPP16b]. Indeed we start the security proof with an hybrid experiment where the simulator of NMZK is used. In this we are guaranteed that the malicious party is behaving honestly by the non-malleability/extractability of NMZK. In the next hybrid experiment we use the simulator of the OT protocol $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ thus extracting the input from the adversary. In the rest of the hybrid experiments we remove the input of the honest party, and use the input extracted via $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to complete the interaction against the adversary. An important difference with the approach used in [GMPP16b] is that in all the steps of our security proof the simulator-extractor of NMZK is used to check every time that the adversary is using the same input in both the executions of the KO protocol even though the adversary is receiving a simulated NMZK of a false statement. More precisely, every time that we change something obtaining a new hybrid experiment, we prove that: 1) the output distributions of the experiments are indistinguishable; 2) the malicious party is behaving honestly (the statement proved by the NMZK given by the adversary is true). We will show that if one of these two invariants does not hold then we can make a reduction that breaks a cryptographic primitive.

The need of a special 4-round OT protocol. Interestingly, the security proof has to address a major issue. After we switch to the simulator of NMZK, we have to change the input of the receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ in some experiment H_i (following the approach used in the security proof of the KO protocol). To demonstrate the indistinguishability between H_i and H_{i-1} we want to rely on the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against a malicious sender. Therefore we construct an adversarial sender $\mathcal{A}_{\mathcal{OT}}$ of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. $\mathcal{A}_{\mathcal{OT}}$ acts as a proxy for the messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ and internally computes the other messages of our protocol. In particular, the 1st and the 3rd rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are given by the challenger (that acts as a receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$), and the 2nd and the 4th messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are given by the malicious party. Furthermore, in order to compute the other messages of our 2PC protocol $\mathcal{A}_{\mathcal{OT}}$ needs to run the simulator-extractor of NMZK, and this requires to rewind from the 3rd to 2nd round. This means that $\mathcal{A}_{\mathcal{OT}}$ needs to complete a 3rd round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$, for every different 2nd round that he receives (this is due to the rewinds made by the simulator of NMZK that are emulated by $\mathcal{A}_{\mathcal{OT}}$). We observe that since the challenger cannot be rewound, $\mathcal{A}_{\mathcal{OT}}$ needs a strategy to answer to these multiple queries w.r.t. $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ without knowing the randomness and the input used by the challenger so far. For these reasons we need $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to enjoy an additional property: the *replayability* of the 3rd round. More precisely, given the messages computed by an honest receiver, the third round can be indistinguishability used to answer to any second round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ sent by a malicious sender. Another issue is that the idea of the security proof explained so far relies on the simulator-extractor of NMZK and this simulator rewinds also from the 4th to the 3rd round. The rewinds made by the simulator-extractor allow a malicious receiver to ask for different 3rd rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. Therefore we need our $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to be also secure against a more powerful malicious receiver that can send multiple (up to a polynomial γ) third rounds to the honest sender. As far as we know the literature does not provide an OT with the properties that we require, so in this work we also provide an OT protocol with these additional features. This clearly is of independent interest.

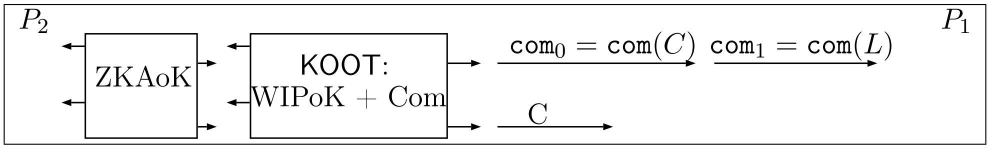

Figure 1: The 4-round KO protocol from trapdoor permutations for functionalities where only one player receives the output.

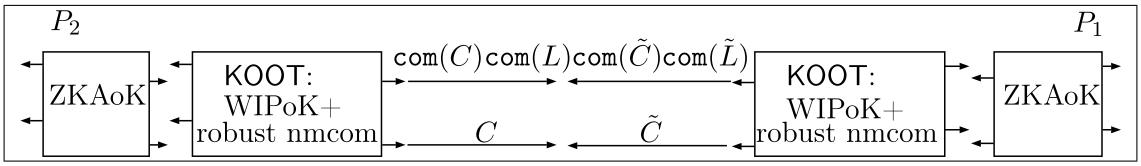

Figure 2: The 4-round protocol of [GMPP16b] for any functionality assuming 3-round 3-robust parallel non-malleable commitments in the simultaneous message exchange model.

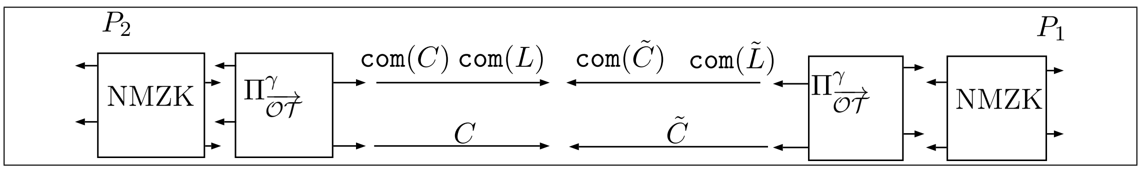

Figure 3: Our 4-round protocol for any functionality assuming trapdoor permutations in the simultaneous message exchange model.

1.2 Special One-Sided Simulatable OT

One of the main building blocks of our 2PC protocol is an OT protocol $\Pi^{\gamma}_{\mathcal{OT}} = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$ one-sided simulatable[2]. Our $\Pi^{\gamma}_{\mathcal{OT}}$ has four rounds where the first (ot_1) and the third (ot_3) rounds are played by the receiver, and the remaining rounds (ot_2 and ot_4) are played by the sender. In addition $\Pi^{\gamma}_{\mathcal{OT}}$ enjoys the following two additional properties.

1. *Replayable third round.* Let $(\mathsf{ot}_1, \mathsf{ot}_2, \mathsf{ot}_3, \mathsf{ot}_4)$ be the messages exchanged by an honest receiver and a malicious sender during an execution of $\Pi^{\gamma}_{\mathcal{OT}}$. For any honestly computed ot'_2, we have that $(\mathsf{ot}_1, \mathsf{ot}_2, \mathsf{ot}_3)$ and $(\mathsf{ot}_1, \mathsf{ot}'_2, \mathsf{ot}_3)$ are identically distributed. Roughly, we are requiring that the third round can be reused in order to answer to any second round ot'_2 sent by a malicious sender.
2. *Repeatability.* We require $\Pi^{\gamma}_{\mathcal{OT}}$ to be secure against a malicious receiver $R^{\star}$ even when the last two rounds of $\Pi^{\gamma}_{\mathcal{OT}}$ can be repeated multiple times. More precisely a 4-round OT protocol that is secure in this setting can be seen as an OT protocol of $2 + 2\gamma$ rounds, with $\gamma \in \{1, \dots, \mathsf{poly}(\lambda)\}$ where λ represents the security parameter. In this protocol $R^{\star}$, upon receiving the 4th round, can continue the execution with $S_{\mathcal{OT}}$ by sending a freshly generated third round of $\Pi^{\gamma}_{\mathcal{OT}}$ up to total of γ 3rd rounds.

[2] In the 2PC protocol we will actually use $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ that roughly corresponds to parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$. More details will be provided later.

Roughly, we require that the output of such $R^\star$ that runs $\Pi^\gamma_{\mathcal{OT}}$ against an honest sender can be simulated by an efficient simulator Sim that has only access to the ideal world functionality $F_{\mathcal{OT}}$ and oracle access to $R^\star$.

The security of $\Pi^\gamma_{\mathcal{OT}}$ is based on the existence of trapdoor permutations[3].

Our techniques. In order to construct $\Pi^\gamma_{\mathcal{OT}}$ we use as a starting point the following basic 3-round semi-honest OT Π_{sh} based on trapdoor permutations (TDPs) of [EGL82, KO04]. Let $l_0, l_1 \in \{0,1\}^\lambda$ be the input of the sender S and b be the input bit of the receiver R.

1. The sender S chooses a trapdoor permutation $(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends f to the receiver R.
2. R chooses $x \leftarrow \{0,1\}^\lambda$ and $z_{1-b} \leftarrow \{0,1\}^\lambda$, computes $z_b = f(x)$ and sends (z_0, z_1).
3. For $c = 0,1$ S computes and sends $w_c = l_c \oplus \mathsf{hc}(f^{-1}(z_c))$

where $\mathsf{hc}(\cdot)$ is a hardcore bit of f. If the parties follow the protocol (i.e. in the semi-honest setting) then S cannot learn the receiver's input (the bit b) as both z_0 and z_1 are random strings. Also, due to the security of the TDP f, R cannot distinguish w_{1-b} from random as long as z_{1-b} is randomly chosen. If we consider a fully malicious receiver $R^\star$ then this protocol is not secure anymore. Indeed $R^\star$ could just compute $z_{1-b} = f(y)$ picking a random $y \leftarrow \{0,1\}^\lambda$. In this way $R^\star$ can retrieve both the inputs of the sender l_0 and l_1. In [KO04] the authors solve this problem by having the parties engaging a coin-flipping protocol such that the receiver is forced to set at least one between z_0 and z_1 to a random string. This is done by forcing the receiver to commit to two strings (r_0, r_1) in the first round (for the coin-flipping) and providing a witness-indistinguishable proof of knowledge (WIPoK) that either $z_0 = r_0 \oplus r'_0$ or $z_1 = r_1 \oplus r'_1$ where r'_0 and r'_1 are random strings sent by the sender in the second round. The resulting protocol, as observed in [ORS15], leaks no information to S about R's input. Moreover the soundness of the WIPoK forces a malicious $R^\star$ to behave honestly, and the PoK allows to extract the input from the adversary in the simulation. Therefore the protocol constructed in [KO04] is one-sided simulatable. Unfortunately this approach is not sufficient to have an OT protocol that has a *replayable* third round. This is due to the to the added WIPoK. More precisely, the receiver has to execute a WIPoK (acting as a prover) in the first three rounds. Clearly, there is no 3-round WIPoK such that given an accepting transcript (a, c, z) one can efficiently compute multiple accepting transcripts w.r.t. different second rounds without knowing the randomness used to compute a. This is the reason why we need to use a different approach in order to construct an OT protocol simulation-based secure against a malicious receiver that also has a replayable 3rd round.

Our construction: $\Pi^\gamma_{\mathcal{OT}}$. We start by considering a trick proposed in [ORS15]. In [ORS15] the authors construct a 4-round black-box OT starting from Π_{sh}. In order to force the receiver to compute a random z_{1-b}, in the first round R sends two commitments c_0 and c_1 such that $c_b = \mathsf{Eqcom}(\cdot), c_{1-b} = \mathsf{Eqcom}(r_{1-b})$. Eqcom is a commitment scheme that is binding if the committer runs the honest committer procedure; however this commitment scheme admits also an indistinguishable equivocal commitment procedure that allows later to open the equivocal commitment

[3]As suggested by Ivan Damgård and Claudio Orlandi in a personal communication, following the approach of [GKM+00], $\Pi^\gamma_{\mathcal{OT}}$ can be also constructed by relying on public key encryption schemes with special properties. More precisely the public key encryption scheme has to be such that that either the ciphertexts can be sampled without knowing the plaintext, or the public key can be sampled without knowing the corresponding secret key. In this paper we give a formal construction and proof only for trapdoor permutations.

as any message. R then proves using a special WIPoK that either c_0 or c_1 is computed using the honest procedure (i.e., at least one of these commitments is binding). Then S in the second round computes $r'_0 \leftarrow \{0,1\}^\lambda$, $r'_1 \leftarrow \{0,1\}^\lambda$ and two TDPs f_0, f_1 with the respective trapdoors and sends (r'_0, r'_1, f_0, f_1) to R. R, upon receiving (r'_0, r'_1, f_0, f_1), picks $x \leftarrow \{0,1\}^\lambda$, computes $r_b = f_b(x) \oplus r'_b$ and sends the opening of c_{1-b} to the message r_{1-b} and the opening of c_b to the message r_b. At this point the sender computes and sends $w_0 = l_0 \oplus \mathsf{hc}(f_0^{-1}(r_0 \oplus r'_0))$, $w_1 = l_1 \oplus \mathsf{hc}(f_1^{-1}(r_1 \oplus r'_1))$. Since at least one between c_0 and c_1 is binding (due to the WIPoK), a malicious receiver can retrieve only one of the sender's input l_b. We observe that this OT protocol is still not sufficient for our propose due to the WIPoK used by the receiver (i.e., the 3rd round is not *replayable*). Moreover we cannot remove the WIPoK otherwise a malicious receiver could compute both c_0 and c_1 using the equivocal procedure thus obtaining l_0 and l_1. Our solution is to replace the WIPoK with some primitives that make replayable the 3rd round, still allowing the receiver to prove that at least one of the commitments sent in the first round is binding. Our key-idea is two use a combination of instance-dependent trapdoor commitment (IDTCom) and non-interactive commitment schemes. An IDTCom is defined over an instance x that could belong to the $\mathcal{NP}$-language L or not. If $x \notin L$ then the IDTCom is perfectly binding, otherwise it is equivocal and the trapdoor information is represented by the witness w for x. Our protocol is described as follows. R sends an IDTCom $\mathtt{tcom}_0$ of r_0 and an IDTCom $\mathtt{tcom}_1$ of r_1. In both cases the instance used is $\mathtt{com}$, a perfectly binding commitment of the bit b (the receiver's input). The $\mathcal{NP}$-language used to compute $\mathtt{tcom}_0$ consists of all valid perfectly binding commitments of the message 0, while the $\mathcal{NP}$-language used to compute $\mathtt{tcom}_1$ consists of all valid perfectly binding commitments of the message 1.

This means that $\mathtt{tcom}_b$ can be opened to any value[4] and $\mathtt{tcom}_{1-b}$ is perfectly binding. It is important to observe that due to the binding property of $\mathtt{com}$ it could be that both $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are binding, but it can never happen that they are both equivocal. Now we can replace the two commitments and the WIPoK used in [ORS15] with $\mathtt{tcom}_0, \mathtt{tcom}_1$ and $\mathtt{com}(b)$ that are sent in the first round. The rest of the protocol stay the same as in [ORS15] with the difference that in the third round the openings to the messages r_0 and r_1 are w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$. What remains to observe is that when a receiver provides a valid third round of this protocol then the same message can be used to answer any second round. Indeed, a well formed third round is accepting if and only if the opening w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are well computed. Therefore whether the third round is accepting or not does not depend on the second round sent by the sender.

Intuitively this protocol is also already secure when we consider a malicious receiver that can send multiple third rounds (up to γ 3rd rounds), thus obtaining an OT protocol of $2 + 2\gamma$ rounds (repeatability). This is because, even though a malicious receiver obtains multiple fourth rounds in response to multiple third rounds sent by $R^\star$, no information about the input of the sender is leaked. Indeed, in our $\Pi^\gamma_{\mathcal{OT}}$, the input of the receiver is fixed in the first round (only one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be equivocal). Therefore the security of the TDP ensures that only l_b can be obtained by $R^\star$ independently of what he does in the third round. In the formal part of the paper we will show that the security of the TDP is enough to deal with such a scenario.

We finally point out that the OT protocol we need has to allow parties to use strings instead of bits as input. More precisely the sender's input is represented by $(l^1_0, l^1_1, \dots, l^m_0, l^m_1)$ where each l^i_b is an λ-bit length string (for $i = 1, \dots, m$ and $b = 0, 1$), while the input of the receiver is λ-bit length string.

This is achieved in two steps. First we construct an OT protocol where the sender's input is

[4] The decommitment information of $\mathtt{com}$ represents the trapdoor of the IDTCom $\mathtt{tcom}_b$.

represented by just two m-bit strings l_0 and l_1 and the receiver's input is still a bit. We obtain this protocol by just using in $\Pi^{\gamma}_{\mathcal{OT}}$ a vector of m hard-core bits instead of just a single hard core bit following the approach of [KO04, GMPP16b]. Then we consider m parallel execution of this modified $\Pi^{\gamma}_{\mathcal{OT}}$ (where the the sender uses a pair of strings as input) thus obtaining $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.

2 Definitions and Tools

2.1 Preliminaries

We denote the security parameter by λ and use "$||$" as concatenation operator (i.e., if a and b are two strings then by $a||b$ we denote the concatenation of a and b). For a finite set Q, $x \leftarrow Q$ denotes a sampling of x from Q with uniform distribution. We use the abbreviation PPT that stands for probabilistic polynomial time. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function.

A *polynomial-time relation* Rel (or *polynomial relation*, in short) is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x, w) in Rel can be decided in time polynomial in $|x|$. For $(x, w) \in \mathsf{Rel}$, we call x the *instance* and w a *witness* for x. For a polynomial-time relation Rel, we define the $\mathcal{NP}$-language L_{Rel} as $L_{\mathsf{Rel}} = \{x | \exists w : (x, w) \in \mathsf{Rel}\}$. Analogously, unless otherwise specified, for an $\mathcal{NP}$-language L we denote by $\mathsf{Rel_L}$ the corresponding polynomial-time relation (that is, $\mathsf{Rel_L}$ is such that $L = L_{\mathsf{Rel_L}}$). We denote by $\hat{L}$ the language that includes both L and all well formed instances that do not have a witness. Moreover we require that membership in $\hat{L}$ can be tested in polynomial time. We implicitly assume that a PPT algorithm that is supposed to receive an instance in $\hat{L}$ will abort immediately if the instance does not belong to $\hat{L}$.

Let A and B be two interactive probabilistic algorithms. We denote by $\langle A(\alpha), B(\beta)\rangle(\gamma)$ the distribution of B's output after running on private input β with A using private input α, both running on common input γ. Typically, one of the two algorithms receives 1^λ as input. A *transcript* of $\langle A(\alpha), B(\beta)\rangle(\gamma)$ consists of the messages exchanged during an execution where A receives a private input α, B receives a private input β and both A and B receive a common input γ. Moreover, we will refer to the *view* of A (resp. B) as the messages it received during the execution of $\langle A(\alpha), B(\beta)\rangle(\gamma)$, along with its randomness and its input. We say that the transcript τ of an execution $b = \langle \mathcal{P}(z), \mathcal{V}\rangle(x)$ is *accepting* if $b = 1$.

We say that a protocol (A, B) is public coin if B sends to A random bits only. When it is necessary to refer to the randomness r used by and algorithm A we use the following notation: $A(\cdot; r)$.

2.2 Standard Definitions

Definition 1 (Proof/argument system). *A pair of* PPT *interactive algorithms* $\Pi = (\mathcal{P}, \mathcal{V})$ *constitutes a* proof system *(resp., an* argument system*) for an* $\mathcal{NP}$*-language* L*, if the following conditions hold:*

Completeness: *For every* $x \in L$ *and* w *such that* $(x, w) \in \mathsf{Rel_L}$*, it holds that:*

$$\text{Prob}\left[\ \langle \mathcal{P}(w), \mathcal{V}\rangle(x) = 1\ \right] = 1.$$

Soundness: *For every interactive (resp.,* PPT *interactive) algorithm* $\mathcal{P}^\star$*, there exists a negligible function* ν *such that for every* $x \notin L$ *and every* z*:*

$$\text{Prob}\left[\ \langle \mathcal{P}^\star(z), \mathcal{V}\rangle(x) = 1\ \right] < \nu(|x|).$$

A proof/argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an $\mathcal{NP}$-language L, enjoys *delayed-input* completeness if $\mathcal{P}$ needs x and w only to compute the last round and $\mathcal{V}$ needs x only to compute the output. Before that, $\mathcal{P}$ and $\mathcal{V}$ run having as input only the size of x. The notion of delayed-input completeness was defined in [CPS+16a].

Definition 2 (Computational indistinguishability). *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles, where X_λ's and Y_λ's are probability distribution over $\{0,1\}^l$, for same $l = \mathsf{poly}(\lambda)$. We say that $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are* computationally indistinguishable*, denoted $X \approx Y$, if for every* PPT *distinguisher $\mathcal{D}$ there exists a negligible function ν such that for sufficiently large $\lambda \in \mathbb{N}$,*

$$\left|\text{Prob}\left[\ t \leftarrow X_\lambda : \mathcal{D}(1^\lambda, t) = 1\ \right] - \text{Prob}\left[\ t \leftarrow Y_\lambda : \mathcal{D}(1^\lambda, t) = 1\ \right]\right| < \nu(\lambda).$$

We note that in the usual case where $|X_\lambda| = \Omega(\lambda)$ and λ can be derived from a sample of X_λ, it is possible to omit the auxiliary input 1^λ. In this paper we also use the definition of *Statistical Indistinguishability*. This definition is the same as Definition 2 with the only difference that the distinguisher $\mathcal{D}$ is unbounded. In this case use $X \equiv_s Y$ to denote that two ensembles are statistically indistinguishable.

Definition 3 (Proof of Knowledge [Dam10]). *A pair $(\mathcal{P}, \mathcal{V})$ of* PPT *interactive algorithms is a* proof of knowledge with knowledge error $k(\cdot)$ *for polynomial-time relation* Rel *if the following properties hold:*

- Completeness. *For every $(x, w) \in \mathsf{Rel}$, it holds that*

$$\text{Prob}\left[\ \langle\mathcal{P}(w), \mathcal{V}\rangle(x) = 1\ \right] = 1.$$

- Knowledge Soundness: *there exists a probabilistic oracle machine* $\mathsf{Extract}$*, called the* extractor*, such that for every interactive machine $\mathcal{P}^\star$ and for every input x accepted by $\mathcal{V}$ when interacting with $\mathcal{P}^\star$ with probability $\epsilon(x) > k(x)$, $\mathsf{Extract}^{\mathcal{P}^\star}(x)$ outputs a witness w for x. Moreover, the expected number of steps performed by $\mathsf{Extract}$ is bounded by $\mathtt{poly}(|x|)/(\epsilon(x) - k(x))$.*

In our security proofs we make use of the following observation. An interactive protocol Π that enjoys the property of completeness and PoK (AoK) is a proof (an argument) system. Indeed suppose by contradiction that is not. By the definition of PoK (AoK) it is possible to extract the witness for every theorem $x \in \{0,1\}^\lambda$ proved by $\mathcal{P}_r^\star$ with probability greater than $\text{Prob}\left[\ \langle\mathcal{P}_r^\star(z), \mathcal{V}\rangle(x) = 1\ \right]$; contradiction.

In this paper we also consider the *adaptive-input* PoK/AoK property for all the protocols that enjoy delayed-input completeness. Adaptive-input PoK/AoK ensures that the PoK/AoK property still holds when a malicious prover can choose the statement adaptively at the last round (see [CPS+16b] for more discussions about adaptive-input PoK/AoK).

Definition 4 (Yao's garbled circuit). *We view Yao's garbled circuit scheme as a tuple of* PPT *algorithms* $(\mathsf{GenGC}, \mathsf{EvalGC})$ *where* GenGC *is the generation procedure which generates a garbled circuit for a circuit* $\mathsf{GC_y}$ *along with labels, and* EvalGC *is the evaluation procedure which evaluates the circuit on the correct labels. Each individual wire i of the circuit is assigned two labels, namely $Z_{i,0}, Z_{i,1}$. More specifically, the two algorithms have the following format:*

- $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F, y)$: GenGC *takes as input a security parameter* λ*, a circuit* F *and a string* $y \in \{0,1\}^\lambda$*. It outputs a garbled circuit* $\mathsf{GC_y}$ *along with the set of all input-wire labels* $\{Z_{1,b}, \dots, Z_{\lambda,b}\}_{b\in\{0,1\}}$*. The garbled circuit may be viewed as representing the function* $F(\cdot, y)$*.*

- $v = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})$: *Given a garbled circuit* $\mathsf{GC_y}$ *and a set of input-wire labels* Z_{i,x_i} *where* $x_i \in \{0,1\}$ *for* $i = 1, \dots, \lambda$, EvalGC *outputs either an invalid symbol* $\perp$*, or a value* $v = F(x, y)$*.*

The following properties are required.

Correctness. $\text{Prob}\,[\, F(x,y) = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})\,] = 1$.

Security. *There exists a* PPT *simulator* SimGC *such that for any* (F, x) *and uniformly random labels* $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$*, it holds that:*

$$(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}) \approx \mathsf{SimGC}(1^\lambda, F, x, v)$$

where $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F, y)$ *and* $v = F(x, y)$*.*

Definition 5 (Trapdoor permutation)**.** *Let* $\mathcal{F}$ *be a triple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *such that if* $\mathsf{Gen}(1^\lambda)$ *outputs a pair* $(f, \mathtt{td})$*, then* $\mathsf{Eval}(f, \cdot)$ *is a permutation over* $\{0,1\}^\lambda$ *and Invert* $(f, \mathtt{td}, \cdot)$ *is its inverse.* $\mathcal{F}$ *is a trapdoor permutation such that for all* PPT *adversaries* $\mathcal{A}$*:*

$$\text{Prob}\left[\, (f, \mathtt{td}) \leftarrow \mathsf{Gen}(1^\lambda); y \leftarrow \{0,1\}^\lambda, x \leftarrow \mathcal{A}(f, y) : \mathsf{Eval}(f, x) = y \,\right] \leq \nu(\lambda).$$

For convenience, we drop $(f, \mathtt{td})$ from the notation, and write $f(\cdot)$, $f^{-1}(\cdot)$ to denote algorithms $\mathsf{Eval}(f, \cdot)$, $\mathsf{Invert}(f, \mathtt{td}, \cdot)$ respectively, when f, $\mathtt{td}$ are clear from the context. Following [KO04, GMPP16b] we assume that $\mathcal{F}$ satisfies (a weak variant of) "certifiability": namely, given some f it is possible to decide in polynomial time whether $\mathsf{Eval}(f, \cdot)$ is a permutation over $\{0,1\}^\lambda$. Let hc be the hardcore bit function for λ bits for the family $\mathcal{F}$. λ hardcore bits are obtained from a single-bit hardcore function h and $f \in \mathcal{F}$ as follows: $\mathsf{hc}(z) = h(z)||h(f(z))||\dots||h(f^{\lambda-1}(z))$. Informally, $\mathsf{hc}(z)$ looks pseudorandom given $f^\lambda(z)$[5].

2.3 Commitment Schemes

Definition 6 (Commitment Scheme)**.** *Given a security parameter* 1^λ*, a commitment scheme* $\mathtt{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *is a two-phase protocol between two* PPT *interactive algorithms, a sender* Sen *and a receiver* Rec*. In the commitment phase* Sen *on input a message* m *interacts with* Rec *to produce a commitment* $\mathtt{com}$*, and the private output* $\mathtt{d}$ *of* Sen*.*

In the decommitment phase, Sen *sends to* Rec *a decommitment information* $(m, \mathtt{d})$ *such that* Rec *accepts* m *as the decommitment of* $\mathtt{com}$*.*

Formally, we say that $\mathtt{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *is a perfectly binding commitment scheme if the following properties hold:*

Correctness:

- *Commitment phase. Let* $\mathtt{com}$ *be the commitment of the message* m *given as output of an execution of* $\mathtt{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *where* Sen *runs on input a message* m*. Let* $\mathtt{d}$ *be the private output of* Sen *in this phase.*

---

[5] $f^\lambda(z)$ means the λ-th iteration of applying f on z.

- *Decommitment phase*[6]. Rec *on input* m *and* $\mathtt{d}$ *accepts* m *as decommitment of* $\mathtt{com}$.

Statistical (resp. Computational) Hiding([Lin10]): *for any adversary (resp.* PPT *adversary)* $\mathcal{A}$ *and a randomly chosen bit* $b \in \{0,1\}$, *consider the following hiding experiment* $\mathsf{ExpHiding}^{b}_{\mathcal{A},\mathsf{CS}}(\lambda)$:

- *Upon input* 1^λ, *the adversary* $\mathcal{A}$ *outputs a pair of messages* m_0, m_1 *that are of the same length.*
- Sen *on input the message* m_b *interacts with* $\mathcal{A}$ *to produce a commitment of* m_b.
- $\mathcal{A}$ *outputs a bit* b' *and this is the output of the experiment.*

For any adversary (resp. PPT *adversary)* $\mathcal{A}$, *there exist a negligible function* ν, *s.t.:*

$$\left|\mathrm{Prob}\left[\ \mathsf{ExpHiding}^{0}_{\mathcal{A},\mathsf{CS}}(\lambda) = 1\ \right] - \mathrm{Prob}\left[\ \mathsf{ExpHiding}^{1}_{\mathcal{A},\mathsf{CS}}(\lambda) = 1\ \right]\right| < \nu(\lambda).$$

Statistical (resp. Computational) Binding: *for every commitment* $\mathtt{com}$ *generated during the commitment phase by a possibly malicious unbounded (resp. malicious* PPT*) sender* $\mathsf{Sen}^\star$ *there exists a negligible function* ν *such that* $\mathsf{Sen}^\star$, *with probability at most* $\nu(\lambda)$, *outputs two decommitments* (m_0, d_0) *and* (m_1, d_1), *with* $m_0 \neq m_1$, *such that* Rec *accepts both decommitments.*

We also say that a commitment scheme is perfectly binding *iff* $\nu(\lambda) = 0$.

When a commitment scheme $(\mathsf{Com}, \mathsf{Dec})$ *is non-interactive, to not overburden the notation, we use the following notation.*

– *Commitment phase.* $(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(m)$ *denotes that* $\mathtt{com}$ *is the commitment of the message* m *and* $\mathtt{dec}$ *represents the corresponding decommitment information.*

– *Decommitment phase.* $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, m) = 1$.

2-Round Instance-Dependent Trapdoor Commitments. Following [COSV17b] here we define a special commitment scheme based on an $\mathcal{NP}$-language L where sender and receiver also receive as input an instance x. While correctness and computational hiding hold for any x, we require that statistical binding holds for $x \notin L$ and moreover knowledge of a witness for $x \in L$ allows to equivocate. Finally, we require that a commitment along with two valid openings to different messages allows to compute the witness for $x \in L$. We recall that $\hat{L}$ denotes the language that includes L and all well formed instances that are not in L.

Definition 7 (2-Round Instance-Dependent Trapdoor Commitments)**.** *Let* 1^λ *be the security parameter,* L *be an* $\mathcal{NP}$*-language and* $\mathsf{Rel_L}$ *be the corresponding* $\mathcal{NP}$*-relation. A triple of* PPT *algorithms* $\mathsf{TC} = (\mathsf{Sen}, \mathsf{Rec}, \mathsf{TFake})$ *is a 2-Round Instance-Dependent Trapdoor Commitment scheme if the following properties hold.*

Correctness. *In the 1st round,* Rec *on input* 1^λ *and* $x \in \hat{L}$ *outputs* ρ. *In the 2nd round* Sen *on input the message* m, 1^λ, ρ *and* $x \in L$ *outputs* $(\mathtt{com}, \mathtt{dec})$. *We will refer to the pair* $(\rho, \mathtt{com})$ *as the commitment of* m. *Moreover we will refer to the execution of the above two rounds including the exchange of the corresponding two messages as the commitment phase. Then* Rec *on input* m, x, $\mathtt{com}$, $\mathtt{dec}$ *and the private coins used to generate* ρ *in the commitment phase outputs 1. We will refer to the execution of this last round including the exchange of*

[6] In this paper we consider only non-interactive commitment and decommitment phase.

$\mathtt{dec}$ *as the decommitment phase. Notice that an adversarial sender* $\mathsf{Sen}^\star$ *could deviate from the behavior of* Sen *when computing and sending* $\mathtt{com}$ *and* $\mathtt{dec}$ *for an instance* $x \in \hat{L}$*. As a consequence* Rec *could output 0 in the decommitment phase. We will say that* $\mathtt{dec}$ *is a valid decommitment of* $(\rho, \mathtt{com})$ *to* m *for an instance* $x \in \hat{L}$*, if* Rec *outputs 1.*

Hiding. *Given a* PPT *adversary* $\mathcal{A}$*, consider the following hiding experiment* $\mathsf{ExpHiding}^b_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ *for* $b = 0, 1$ *and* $x \in \hat{L}_R$*:*

- *On input* 1^λ *and* x*,* $\mathcal{A}$ *outputs a message* m*, along with* ρ*.*
- *The challenger on input* x, m, ρ, b *works as follows: if* $b = 0$ *then it runs* Sen *on input* m*,* x *and* ρ*, obtaining a pair* $(\mathtt{com}, \mathtt{dec})$*, otherwise it runs* TFake *on input* x *and* ρ*, obtaining a pair* $(\mathtt{com}, \mathtt{aux})$*. The challenger outputs* $\mathtt{com}$*.*
- $\mathcal{A}$ *on input* $\mathtt{com}$ *outputs a bit* b' *and this is the output of the experiment.*

We say that hiding *holds if for any* PPT *adversary* $\mathcal{A}$ *there exist a negligible function* ν*, s.t.:*

$$\left|\mathrm{Prob}\left[\ \mathsf{ExpHiding}^0_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] - \mathrm{Prob}\left[\ \mathsf{ExpHiding}^1_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right]\right| < \nu(\lambda).$$

Special Binding. *There exists a* PPT *algorithm that on input a commitment* $(\rho, \mathtt{com})$*, the private coins used by* Rec *to compute* ρ*, and two valid decommitments* $(\mathtt{dec}, \mathtt{dec}')$ *of* $(\rho, \mathtt{com})$ *to two different messages* m *and* m'*, outputs* w *s.t.* $(x, w) \in \mathsf{Rel_L}$ *with overwhelming probability.*

Instance-Dependent Binding. *For every malicious unbounded sender* $\mathsf{Sen}^\star$ *there exists a negligible function* ν *s.t. for a commitment* $(\rho, \mathtt{com})$ $\mathsf{Sen}^\star$*, with probability at most* $\nu(\lambda)$*, outputs two decommitments* (m_0, d_0) *and* (m_1, d_1) *with* $m_0 \neq m_1$ *s.t.* Rec *on input the private coins used to compute* ρ *and* $x \notin L$ *accepts both decommitments.*

Trapdoorness. *For any* PPT *adversary* $\mathcal{A}$ *there exist a negligible function* ν*, s.t. for all* $x \in L$ *it holds that:*

$$\left|\mathrm{Prob}\left[\ \mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] - \mathrm{Prob}\left[\ \mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right]\right| < \nu(\lambda)$$

where $\mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ *and* $\mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ *are defined below*[7]*.*

$\mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$*:*	$\mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$*:*
-On input 1^λ *and* x*,* $\mathcal{A}$ *outputs* (ρ, m)*.*	*-On input* 1^λ *and* x*,* $\mathcal{A}$ *outputs* (ρ, m)*.*
-Sen *on input* 1^λ*,* x*,* m *and* ρ*, outputs* $(\mathtt{com}, \mathtt{dec})$*.*	-TFake *on input* 1^λ*,* x *and* ρ*, outputs* $(\mathtt{com}, \mathtt{aux})$*.*
	-TFake *on input* $\mathtt{tk}$ *s.t.* $(x, \mathtt{tk}) \in \mathsf{Rel_L}$*,* x*,* ρ*,* $\mathtt{com}$*,* $\mathtt{aux}$ *and* m *outputs* $\mathtt{dec}$*.*
-$\mathcal{A}$ *on input* $(\mathtt{com}, \mathtt{dec})$ *outputs a bit* b *and this is the output of the experiment.*	-$\mathcal{A}$ *on input* $(\mathtt{com}, \mathtt{dec})$ *outputs a bit* b *and this is the output of the experiment.*

In this paper we consider also a non-interactive version of Instance-Dependent Trapdoor Commitments. The only difference in the definition is that the first round sent by the receiver to the sender just disappears. In this case we use the following simplified notation.

[7]We assume wlog that $\mathcal{A}$ is stateful.

- *Commitment phase.* $(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Sen}(m, 1^\lambda, x)$ *denotes that* $\mathtt{com}$ *is the commitment of the message* m *and* $\mathtt{dec}$ *represents the corresponding decommitment information.*
- *Decommitment phase.* $1 \leftarrow \mathsf{Rec}(m, x, \mathtt{com}, \mathtt{dec})$*.*
- *Trapdoor algorithms.* $(\mathtt{com}, \mathtt{aux}) \leftarrow \mathsf{TFake}(1^\lambda, x)$*,* $\mathtt{dec} \leftarrow \mathsf{TFake}(\mathtt{tk}, x, \mathtt{com}, \mathtt{aux}, m)$ *with* $(x, \mathtt{tk}) \in \mathsf{Rel_L}$*.*

In the rest of the work, we say that the sender uses the *honest procedure* when he computes the commitment $\mathtt{com}$ of a message m along with the decommitment information $\mathtt{dec}$ running Sen. Instead, the sender uses *trapdoor procedure* when he computes $\mathtt{com}$ and $\mathtt{dec}$ running TFake.

We recall that, as has been observed in [COSV17b], a non-interactive Instance-Dependent Trapdoor Commitment scheme can be instantiated by one-to-one OWFs.

2.4 Delayed-Input Non-Malleable Zero Knowledge

Here we follow [COSV17a]. The definition of [COSV17a] allows the adversary to explicitly select the statement, and as such the adversary provides also the witness for the prover. The simulated game however will filter out the witness so that the simulator will receive only the instance. This approach strictly follows the one of [SCO+01] where adaptive-input selection is explicitly allowed and managed in a similar way. As final remark, this definition will require the existence of a black-box simulator since a non-black-box simulator could retrieve from the code of the adversary the witness for the adaptively generated statement. The non-black-box simulator could then run the honest prover procedure, therefore canceling completely the security flavor of the simulation paradigm.

Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a delayed-input interactive argument system for a $\mathcal{NP}$-language L with witness relation $\mathsf{Rel_L}$. Consider a PPT MiM adversary $\mathcal{A}$ that is simultaneously participating in one left session and $\mathsf{poly}(\lambda)$ right sessions. Before the execution starts, $\mathcal{P}, \mathcal{V}$ and $\mathcal{A}$ receive as a common input the security parameter in unary 1^λ. Additionally $\mathcal{A}$ receives as auxiliary input $z \in \{0,1\}^\star$. In the left session $\mathcal{A}$ verifies the validity of the prove given by $\mathcal{P}$ with respect to the statement x (chosen adaptively in the last round of Π). In the right sessions $\mathcal{A}$ proves the validity of the statements $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$[8] (chosen adaptively in the last round of Π) to the honest verifiers $\mathcal{V}_1, \dots, \mathcal{V}_{\mathsf{poly}(\lambda)}$.

More precisely in the left session $\mathcal{A}$, before the last round of Π is executed, adaptively selects the statement x to be proved and the witness w, s.t. $(x, w) \in \mathsf{Rel_L}$, and sends them to $\mathcal{P}$.

Let $\mathsf{View}^{\mathcal{A}}(1^\lambda, z)$ denote a random variable that describes the view of $\mathcal{A}$ in the above experiment.

Definition 8 (Delayed-input NMZK)**.** *A delayed-input argument system* $\Pi = (\mathcal{P}, \mathcal{V})$ *for an* $\mathcal{NP}$*-language* L *with witness relation* $\mathsf{Rel_L}$ *is delayed-input non-malleable zero knowledge (NMZK) if for any MiM adversary* $\mathcal{A}$ *that participates in one left session and* $\mathsf{poly}(\lambda)$ *right sessions, there exists a expected* PPT *machine* $S(1^\lambda, z)$ *such that:*

1. *Let* $(\mathsf{View}, w_1, \dots, w_{\mathsf{poly}(\lambda)})$ *denote the output of* $S(1^\lambda, z)$*, for some* $z \in \{0,1\}^\star$*. The probability ensembles* $\{S^1(1^\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^\star}$ *and* $\{\mathsf{View}^{\mathcal{A}}(1^\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^\star}$ *are computationally indistinguishable over* λ*, where* $S^1(1^\lambda, z)$ *denotes the first output of* $S(1^\lambda, z)$*.*
2. *For every* $i \in \{1, \dots, \mathsf{poly}(\lambda)\}$*, if the* i*-th right session is accepting w.r.t. some statement* x_i *and* $\mathcal{A}$ *does not acts as a proxy (by simply sending back and forward the massages of the left*

[8] We denote (here and in the rest of the paper) by $\tilde{\delta}$ a value associated with the right session where δ is the corresponding value in the left session.

session), then w_i *is s.t.* $(x_i, w_i) \in \mathsf{Rel_L}$[9].

The above definition of NMZK allows the adversary to select statements adaptively in the last round both in left and in the right sessions. Therefore any argument system that is NMZK according to the above definition enjoys also adaptive-input argument of knowledge.

2.5 Two-party Computation with a Simultaneous Message Exchange Channel

Our Two-Party Computation (2PC) protocol is secure in the same model used in [GMPP16b, Pol16], therefore the following definition is taken almost verbatim from [GMPP16b, Pol16].

A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ where $F = (F_1, F_2)$. That is, for every pair of inputs (x, y), the output-pair is a random variable $(F_1(x, y), F_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $F_1(x, y)$ and the second party (with input y) wishes to obtain $F_2(x, y)$.

Adversarial behaviour. Loosely speaking, the aim of a secure two-party protocol is to protect an honest party against dishonest behaviour by the other party. In this paper, we consider malicious adversaries who may arbitrarily deviate from the specified protocol. When considering malicious adversaries, there are certain undesirable actions that cannot be prevented. Specifically, a party may refuse to participate in the protocol, may substitute its local input (and use instead a different input) and may abort the protocol prematurely. One ramification of the adversary's ability to abort, is that it is impossible to achieve fairness. That is, the adversary may obtain its output while the honest party does not. In this work we consider a static corruption model, where one of the parties is adversarial and the other is honest, and this is fixed before the execution begins.

Communication channel. In our result we consider a secure simultaneous message exchange channel in which all parties can simultaneously send messages over the channel at the same communication round but allowing a rushing adversary. Moreover, we assume an asynchronous network[10] where the communication is open and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

Execution in the ideal model. An ideal execution proceeds as follows. Each party obtains an input, denoted w ($w = x$ for P_1, and $w = y$ for P_2). An honest party always sends w to the trusted party. A malicious party may, depending on w, either abort or send some $w' \in \{0,1\}^{|w|}$ to the trusted party. In case it has obtained an input pair (x, y), the trusted party first replies to the first party with $F_1(x, y)$. Otherwise (i.e., in case it receives only one valid input), the trusted party replies to both parties with a special symbol $\perp$. In case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it $\perp$ after receiving its output. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $F_2(x, y)$ to the second party. Outputs: an honest

[9] In this definition we do not consider identities, since we do not need them for our propose of constructing a 2PC protocol.

[10] The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the message obtained from the trusted party.

Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a functionality where $F = (F_1, F_2)$ and let $S = (S_1, S_2)$ be a pair of non-uniform probabilistic expected polynomial-time machines (representing parties in the ideal model). Such a pair is admissible if for at least one $i \in \{0,1\}$ we have that S_i is honest (i.e., follows the honest party instructions in the above-described ideal execution). Then, the joint execution of F under S in the ideal model (on input pair (x, y) and security parameter λ), denoted $\mathsf{IDEAL}_{F,S(z)}(1^\lambda, x, y)$ is defined as the output pair of S_1 and S_2 from the above ideal execution.

Execution in the real model. We next consider the real model in which a real (two-party) protocol is executed (and there exists no trusted third party). In this case, a malicious party may follow an arbitrary feasible strategy; that is, any strategy implementable by non-uniform probabilistic polynomial-time machines. In particular, the malicious party may abort the execution at any point in time (and when this happens prematurely, the other party is left with no output). Let F be as above and let Π be a two-party protocol for computing F. Furthermore, let $A = (A_1, A_2)$ be a pair of non-uniform probabilistic polynomial-time machines (representing parties in the real model). Such a pair is admissible if for at least one $i \in \{0,1\}$ we have that A_i is honest (i.e., follows the strategy specified by Π). Then, the joint execution of Π under A in the real model, denoted $\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda)$, is defined as the output pair of A_1 and A_2 resulting from the protocol interaction.

Definition 9 (secure two-party computation)**.** *Let F and Π be as above. Protocol Π is said to securely compute F (in the malicious model) if for every pair of admissible non-uniform probabilistic polynomial-time machines $A = (A_1, A_2)$ that run with auxiliary input z for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines $S = (S_1, S_2)$ (that use z as auxiliary input) for the ideal model, such that:*

$$\{\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, x, y)\}_{\lambda\in\mathbb{N},z,x,y\in\{0,1\}^\star} \approx \{\mathsf{IDEAL}_{f,S(z)}(1^\lambda, x, y)\}_{\lambda\in\mathbb{N},z,x,y\in\{0,1\}^\star}.$$

We note that the above definition assumes that the parties know the input lengths (this can be seen from the requirement that $|x| = |y|$). Some restriction on the input lengths is unavoidable, see Section 7.1 of [Gol04] for discussion. We also note that we allow the ideal adversary/simulator to run in expected (rather than strict) polynomial-time. This is essential for constant-round protocols.

2.6 Oblivious Transfer

Here we follow [ORS15]. Oblivious Transfer (OT) is a two-party functionality $F_{\mathcal{OT}}$, in which a sender S holds a pair of strings (l_0, l_1), and a receiver R holds a bit b, and wants to obtain the string l_b. The security requirement for the $F_{\mathcal{OT}}$ functionality is that any malicious receiver does not learn anything about the string l_{1-b} and any malicious sender does not learn which string has been transferred. This security requirement is formalized via the ideal/real world paradigm. In the ideal world, the functionality is implemented by a trusted party that takes the inputs from S and R and provides the output to R and is therefore secure by definition. A real world protocol Π securely realizes the ideal $F_{\mathcal{OT}}$ functionalities, if the following two conditions hold. (a) Security against a malicious receiver: the output of any malicious receiver $R^\star$ running one execution of Π

Functionality $F_{\mathcal{OT}}$

$F_{\mathcal{OT}}$ running with a sender S a receiver R and an adversary Sim proceeds as follows:

- Upon receiving a message $(\mathsf{send}, l_0, l_1, S, R)$ from S where each $l_0, l_1 \in \{0,1\}^\lambda$, record the tuple (l_0, l_1) and send send to R and Sim. Ignore any subsequent send messages.
- Upon receiving a message $(\mathsf{receive}, b)$ from R, where $b \in \{0,1\}$ send l_b to R and $\mathsf{receive}$ to S and Sim and halt. (If no $(\mathsf{send}, \cdot)$ message was previously sent, do nothing).

Figure 4: The Oblivious Transfer Functionality $F_{\mathcal{OT}}$.

with an honest sender S can be simulated by a PPT simulator Sim that has only access to the ideal world functionality $F_{\mathcal{OT}}$ and oracle access to $R^\star$. (b) Security against a malicious sender. The joint view of the output of any malicious sender $S^\star$ running one execution of Π with R and the output of R can be simulated by a PPT simulator Sim that has only access to the ideal world functionality functionality $F_{\mathcal{OT}}$ and oracle access to $S^\star$. In this paper we consider a weaker definition of $F_{\mathcal{OT}}$ that is called one-sided simulatable $F_{\mathcal{OT}}$, in which we do not demand the existence of a simulator against a malicious sender, but we only require that a malicious sender cannot distinguish whether the honest receiver is playing with bit 0 or 1. A bit more formally, we require that for any PPT malicious sender $S^\star$ the view of $S^\star$ executing Π with the R playing with bit 0 is computationally indistinguishable from the view of $S^\star$ where R is playing with bit 1. Finally, we consider the $F^m_{\mathcal{OT}}$ functionality where the sender S and the receiver R run m executions of OT in parallel. The formal definitions of one-sided secure $F_{\mathcal{OT}}$ and one-sided secure $F^m_{\mathcal{OT}}$ follow.

Definition 10 ([ORS15])**.** *Let $F_{\mathcal{OT}}$ be the Oblivious Transfer functionality as shown in Fig. 4. We say that a protocol Π securely computes $F_{\mathcal{OT}}$ with one-sided simulation if the following holds:*

1. *For every non-uniform PPT adversary $R^\star$ controlling the receiver in the real model, there exists a non-uniform PPT adversary Sim for the ideal model such that*

$$\{\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda} \approx \mathsf{IDEAL}_{F_{\mathcal{OT}},\mathsf{Sim}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda}$$

   where $\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)$ denotes the distribution of the output of the adversary $R^\star$ (controlling the receiver) after a real execution of protocol Π, where the sender S has inputs l_0, l_1 and the receiver has input b. $\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(1^\lambda)$ denotes the analogous distribution in an ideal execution with a trusted party that computes $F_{\mathcal{OT}}$ for the parties and hands the output to the receiver.

2. *For every non-uniform PPT adversary $S^\star$ controlling the sender it holds that:*

$$\{\mathsf{View}^R_{\Pi,S^\star(z)}(l_0, l_1, 0)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{View}^R_{\Pi,S^\star(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^\star}$$

   where $\mathsf{View}^R_{\Pi,S^\star(z)}$ denotes the view of adversary $S^\star$ after a real execution of protocol Π with the honest receiver R.

Definition 11 (Parallel oblivious transfer functionality $F^m_{\mathcal{OT}}$ [ORS15])**.** *The parallel Oblivious Transfer Functionality $F^m_{\mathcal{OT}}$ is identical to the functionality $F_{\mathcal{OT}}$, with the difference that takes* *in input m pairs of string from S $(l_0^1, l_1^1, \dots, l_0^m, l_1^m)$ (whereas $F_{\mathcal{OT}}$ takes just one pair of strings from S) and m bits from R, $b_1, \dots, b_m$ (whereas $F_{\mathcal{OT}}$ takes one bit from R) and outputs to the receiver values $(l_{b_1}^1, \dots, l_{b_m}^m)$ while the sender receives nothing.*

Definition 12 ([ORS15]). *Let $F_{\mathcal{OT}}^m$ be the Oblivious Transfer functionality as described in Def. 11. We say that a protocol Π securely computes $F_{\mathcal{OT}}^m$ with one-sided simulation if the following holds:*

1. *For every non-uniform* PPT *adversary $R^\star$ controlling the receiver in the real model, there exists a non-uniform* PPT *adversary* Sim *for the ideal model such that for every $x_1 \in \{0,1\}, \dots, x_m \in \{0,1\}$*

$$\{\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda, (l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\} \approx \\ \mathsf{IDEAL}_{F_{\mathcal{OT}}^m,\mathsf{Sim}(z)}(1^\lambda), (l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\}_{z\in\{0,1\}^\lambda}$$

   where $\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)$ denotes the distribution of the output of the adversary $R^\star$ (controlling the receiver) after a real execution of protocol Π, where the sender S has inputs $(l_0^1, l_1^1, \dots, l_0^m, l_1^m)$ and the receiver has input $(x_1, \dots, x_m)$. $\mathsf{IDEAL}_{F_{\mathcal{OT}}^m,\mathsf{Sim}(z)}(1^\lambda)$ denotes the analogous distribution in an ideal execution with a trusted party that computes $F_{\mathcal{OT}}^m$ for the parties and hands the output to the receiver.

2. *For every non-uniform* PPT *adversary $S^\star$ controlling the sender it holds that for every $x_1 \in \{0,1\}, \dots, x_m \in \{0,1\}$ and for every $y_1 \in \{0,1\}, \dots, y_m \in \{0,1\}$:*

$$\{\mathsf{View}^R_{\Pi,S^\star(z)}((l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\}_{z\in\{0,1\}^\star} \approx \\ \{\mathsf{View}^R_{\Pi,S^\star(z)}((l_0^1, l_1^1, \dots, l_0^m, l_1^m), (y_1, \dots, y_m))\}_{z\in\{0,1\}^\star}$$

   where $\mathsf{View}^R_{\Pi,S^\star(z)}$ denotes the view of adversary $S^\star$ after a real execution of protocol Π with the honest receiver R.

We remark that in this notions of OT we do not suppose the existence of a simultaneous message exchange channel.

3 Our OT Protocol $\Pi_{\mathcal{OT}}^\gamma = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$

We use the following tools.

1. A non-interactive perfectly binding, computationally hiding commitment scheme $\mathsf{PBCOM} = (\mathsf{Com}, \mathsf{Dec})$.
2. A trapdoor permutation $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$[11] with the hardcore bit function for λ bits $\mathsf{hc}(\cdot)$ (see Def. 5).
3. A non-interactive IDTC scheme $\mathsf{TC}_0 = (\mathsf{Sen}_0, \mathsf{Rec}_0, \mathsf{TFake}_0)$ for the $\mathcal{NP}$-language $L_0 = \{\mathtt{com} : \exists\ \mathtt{dec}$ s.t. $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, 0) = 1\}$.
4. A non-interactive IDTC scheme $\mathsf{TC}_1 = (\mathsf{Sen}_1, \mathsf{Rec}_1, \mathsf{TFake}_1)$ for the $\mathcal{NP}$-language $L_1 = \{\mathtt{com} : \exists\ \mathtt{dec}$ s.t. $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, 1) = 1\}$.

[11] We recall that for convenience, we drop $(f, \mathtt{td})$ from the notation, and write $f(\cdot)$, $f^{-1}(\cdot)$ to denote algorithms $\mathsf{Eval}(f, \cdot)$, $\mathsf{Invert}(f, \mathtt{td}, \cdot)$ respectively, when f, $\mathtt{td}$ are clear from the context. Also we omit the generalization to a family of TDPs.

Let $b \in \{0,1\}$ be the input of $R_{\mathcal{OT}}$ and $l_0, l_1 \in \{0,1\}^\lambda$ be the input of $S_{\mathcal{OT}}$, we now give the description of our protocol following Fig. 5.

In the **first round** $R_{\mathcal{OT}}$ runs Com on input the message to be committed b in order to obtain the pair $(\mathtt{com}, \mathtt{dec})$. On input the instance $\mathtt{com}$ and a random string r^1_{b-1}, $R_{\mathcal{OT}}$ runs Sen_{1-b} in order to compute the pair $(\mathtt{tcom}_{1-b}, \mathtt{tdec}_{1-b})$. We observe that the *Instance-Dependent Binding* property of the IDTCs, the description of the $\mathcal{NP}$-language L_{1-b} and the fact that in $\mathtt{com}$ the bit b has been committed, ensure that $\mathtt{tcom}_{1-b}$ can be opened only to the value r^1_{b-1}.[12] $R_{\mathcal{OT}}$ runs the trapdoor procedure of the IDTC scheme TC_b. More precisely $R_{\mathcal{OT}}$ runs TFake_b on input the instance $\mathtt{com}$ to compute the pair $(\mathtt{tcom}_b, \mathtt{aux})$. In this case $\mathtt{tcom}_b$ can be equivocated to any message using the trapdoor (the opening information of $\mathtt{com}$), due to the trapdoorness of the IDTC, the description of the $\mathcal{NP}$-language L_b and the message committed in $\mathtt{com}$ (that is represented by the bit b). $R_{\mathcal{OT}}$ sends $\mathtt{tcom}_0$, $\mathtt{tcom}_1$ and $\mathtt{com}$ to $S_{\mathcal{OT}}$.

In the **second round** $S_{\mathcal{OT}}$ picks two random strings R_0, R_1 and two trapdoor permutations $(f_{0,1}, f_{1,1})$ along with their trapdoors $(f^{-1}_{0,1}, f^{-1}_{1,1})$. Then $S_{\mathcal{OT}}$ sends R_0, R_1, $f_{0,1}$ and $f_{1,1}$ to $R_{\mathcal{OT}}$.

In the **third round** $R_{\mathcal{OT}}$ checks whether or not $f_{0,1}$ and $f_{1,1}$ are valid trapdoor permutations. In the negative case $R_{\mathcal{OT}}$ aborts, otherwise $R_{\mathcal{OT}}$ continues with the following steps. $R_{\mathcal{OT}}$ picks a random string z'_1 and computes $z_1 = f_{b,1}(z'_1)$. $R_{\mathcal{OT}}$ now computes $r^1_b = z_1 \oplus R_b$ and runs TFake_b on input $\mathtt{dec}$, $\mathtt{com}$, $\mathtt{tcom}_b$, $\mathtt{aux}$ and r^1_b in order to obtain the equivocal opening $\mathtt{tdec}_b$ of the commitment $\mathtt{tcom}_b$ to the message r^1_b. $R_{\mathcal{OT}}$ renames r_b to r^1_b and $\mathtt{tdec}_b$ to $\mathtt{tdec}^1_b$ and sends to $S_{\mathcal{OT}}$ $(\mathtt{tdec}^1_0, r^1_0)$ and $(\mathtt{tdec}^1_1, r^1_1)$.

In the **fourth round** $S_{\mathcal{OT}}$ checks whether or not $(\mathtt{tdec}^1_0, r^1_0)$ and $(\mathtt{tdec}^1_1, r^1_1)$ are valid openings w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$. In the negative case $S_{\mathcal{OT}}$ aborts, otherwise $S_{\mathcal{OT}}$ computes $W^1_0 = l_0 \oplus \mathsf{hc}(f^{-\lambda}_{0,1}(r^1_0 \oplus R_0))$ and $W^1_1 = l_1 \oplus \mathsf{hc}(f^{-\lambda}_{1,1}(r^1_1 \oplus R_1))$. Informally $S_{\mathcal{OT}}$ encrypts his inputs l_0 and l_1 through a one-time pad using as a secret key the pre-image of $r^1_0 \oplus R_0$ for l_0 and the pre-image of $r^1_1 \oplus R_1$ for l_1. $S_{\mathcal{OT}}$ also computes two trapdoor permutations $(f_{0,2}, f_{1,2})$ along with their trapdoors $(f^{-1}_{0,2}, f^{-1}_{1,2})$ and sends $(W^1_0, W^1_1, f_{0,2}, f_{1,2})$ to $R_{\mathcal{OT}}$. At this point the third and the fourth rounds are repeated up to $\gamma - 1$ times using fresh randomness as showed in Fig. 5. In the last round no trapdoor permutations are needed/sent.

In the **output phase**, $R_{\mathcal{OT}}$ computes and outputs $l_b = W^1_b \oplus \mathsf{hc}(z'_1)$. That is, $R_{\mathcal{OT}}$ just uses the information gained in the fourth round to compute the output. It is important to observe that $R_{\mathcal{OT}}$ can correctly and efficiently compute the output because $z' = r^1_b \oplus R_b$. Moreover $R_{\mathcal{OT}}$ cannot compute l_{1-b} because he has no way to change the value committed in $\mathtt{tcom}_{1-b}$ and invert the TDP since it is suppose to be hard without having the trapdoor.

We observe that a malicious sender $S^\star_{\mathcal{OT}}$ could easily understand the input bit of $R_{\mathcal{OT}}$ when $\gamma > 1$. This is not a problem since for our application we need to prove the security of $\Pi^\gamma_{\mathcal{OT}}$ to hold against malicious sender only for $\gamma = 1$. We only consider $\gamma = \mathsf{poly}(\lambda)$ when proving the security of $\Pi^\gamma_{\mathcal{OT}}$ against malicious receiver.

In order to construct our protocol for two-party computation in the simultaneous message exchange model we need to consider an extended version of $\Pi^\gamma_{\mathcal{OT}}$, that we denote by $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$. In $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ the $S_{\overrightarrow{\mathcal{OT}}}$'s input is represented by m pairs $(l^1_0, l^1_1, \dots, l^m_0, l^m_1)$ and the $R_{\overrightarrow{\mathcal{OT}}}$'s input is represented by the sequence $b_1, \dots, b_m$ with $b_i \in \{0,1\}$ for all $i = 1, \dots, m$. In this case the output of $R_{\overrightarrow{\mathcal{OT}}}$ is $(l_{b_1}, \dots, l_{b_m})$. We construct $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$ by simply considering m parallel iterations of $\Pi^\gamma_{\mathcal{OT}}$ and then we prove that it securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation (see

[12] $\mathtt{com}$ does not belong to the $\mathcal{NP}$-language L_{b-1}, therefore $\mathtt{tcom}_{1-b}$ is a perfectly binding commitment.

$R_{\mathcal{OT}}(b)$		$S_{\mathcal{OT}}(l_0, l_1)$
$(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(1^\lambda, b)$; $(\mathtt{tcom}_b, \mathtt{aux}) \leftarrow \mathsf{TFake}_b(1^\lambda, \mathtt{com})$; $r_{1-b} \leftarrow \{0,1\}^\lambda$; $(\mathtt{tcom}_{1-b}, \mathtt{tdec}_{1-b}) \leftarrow \mathsf{Sen}_{1-b}(1^\lambda, r_{1-b}, \mathtt{com})$.		
	$\xrightarrow{\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1}$	
		$R_0 \leftarrow \{0,1\}^\lambda$; $R_1 \leftarrow \{0,1\}^\lambda$; $(f_{0,1}, f_{0,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$; $(f_{1,1}, f_{1,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$.
	$\xleftarrow{R_0, R_1, f_{0,1}, f_{1,1}}$	
$z_1' \leftarrow \{0,1\}^\lambda$; $z_1 = f_{b,1}^\lambda(z_1')$; $r_b^1 = z_1 \oplus R_b$; $\mathtt{tdec}_b^1 \leftarrow \mathsf{TFake}_b(\mathtt{dec}, \mathtt{com}, \mathtt{tcom}_b, \mathtt{aux}, r_b^1)$; $\mathtt{tdec}_{1-b}^1 = \mathtt{tdec}_{1-b}$, $r_{1-b}^1 = r_{1-b}$.	$\xrightarrow{(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)}$	
		$(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$; $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$; $W_0^1 = l_0 \oplus \mathsf{hc}(f_{0,1}^{-\lambda}(r_0^1 \oplus R_0))$; $W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$.
	$\xleftarrow{W_0^1, W_1^1, f_{0,2}, f_{1,2}}$	
$z_2' \leftarrow \{0,1\}^\lambda$; $z_2 = f_{b,2}^\lambda(z_2')$; $r_b^2 = z_2 \oplus R_b$; $\mathtt{tdec}_b^2 \leftarrow \mathsf{TFake}_b(\mathtt{dec}, \mathtt{com}, \mathtt{tcom}_b, \mathtt{aux}, r_b^2)$; $\mathtt{tdec}_{1-b}^2 = \mathtt{tdec}_{1-b}$, $r_{1-b}^2 = r_{1-b}$.	$\xrightarrow{(\mathtt{tdec}_0^2, r_0^2), (\mathtt{tdec}_1^2, r_1^2)}$	
		$(f_{0,3}, f_{0,3}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$; $(f_{1,3}, f_{1,3}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$; $W_0^2 = l_0 \oplus \mathsf{hc}(f_{0,2}^{-\lambda}(r_0^2 \oplus R_0))$; $W_1^2 = l_1 \oplus \mathsf{hc}(f_{1,2}^{-\lambda}(r_1^2 \oplus R_1))$.
	$\xleftarrow{W_0^2, W_1^2, f_{0,3}, f_{1,3}}$	
	$\vdots$	
	$\xrightarrow{(\mathtt{tdec}_0^\gamma, r_0^\gamma), (\mathtt{tdec}_1^\gamma, r_1^\gamma)}$	
	$\xleftarrow{W_0^\gamma, W_1^\gamma}$	
Output $l_b = W_b^1 \oplus \mathsf{hc}(z_1')$.		

Figure 5: Description of $\Pi_{\mathcal{OT}}^\gamma$.

Definition 12).

Proof sketch. The security proof of $\Pi^{\gamma}_{\mathcal{OT}}$ is divided in two parts. In the former we prove the security against a malicious sender with $\gamma = 1$ and in the latter we prove the security of $\Pi^{\gamma}_{\mathcal{OT}}$ against a malicious receiver with $\gamma = \mathsf{poly}(\lambda)$. In order to prove the security against malicious sender we recall that for the definition of one-sided simulation it is just needed the no information about R's input is leaked to $S^{\star}$. We consider the experiment H_0 where R's input is 0 and the experiment H_1 where R's input is 1 and we prove that $S^{\star}$ cannot distinguish between H_0 and H_1. More precisely we consider the experiment H^a where $\mathtt{tcom}_0$ and the corresponding opening is computed without using the trapdoor (the randomness of $\mathtt{com}$) and relying on the trapdoorness of the IDTCom TC_0 we prove that $H_0 \approx H^a$. Then we consider the experiment H^b where the value committed in $\mathtt{com}$ goes from 0 to 1 and prove that $H^a \approx H^b$ due to the hiding of $\mathtt{com}$. We observe that this reduction can be made because to compute both H^a and H^b the opening informations of $\mathtt{com}$ are not required anymore. The proof ends with the observation that $H^b \approx H_1$ due to the trapdoorness of the IDTCom TC_1.

To prove the security against a malicious receiver $R^{\star}$ we need to show a simulator Sim. Sim rewinds $R^{\star}$ from the third to the second round by sending every time freshly generated R_0 and R_1. Sim then checks whether the values r_0^1 and r_1^1 change during the rewinds. We recall that $\mathtt{com}$ is a perfectly binging commitment, therefore only one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be opened to multiple values using the trapdoor procedure ($\mathtt{com}$ can belong only to one of the $\mathcal{NP}$-languages L_0 and L_1). Moreover, intuitively, the only way that $R^{\star}$ can compute the output is by equivocating one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ based on the values R_0, R_1 received in the second round. This means that if during the rewinds the value opened w.r.t. $\mathtt{tcom}_b$ changes, then the input that $R^{\star}$ is using is b. Therefore the simulator can call the ideal functionality thus obtaining l_b. At this point Sim uses l_b to compute W_b^1 according to the description of $\Pi^{\gamma}_{\mathcal{OT}}$ and sets W_{1-b}^1 to a random string. Moreover Sim will use the same strategy used to compute W_b^1 and W_{1-b}^1 to compute, respectively W_b^i and W_{1-b}^i for $i = 2, \dots, \gamma$. In case during the rewinds the value r_0^1, r_1^1 stay the same, then Sim sets both W_0^1 and W_1^1 to random strings. We observe that $R^{\star}$ could detect that now W_0^1 and W_1^1 are computed in a different way, but this would violate the security of the TDPs.

Theorem 1. *Assuming TDPs, for any $\gamma > 0$ $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation. Moreover the third round is* replayable.

Proof. We first observe that in third round of $\Pi^{\gamma}_{\mathcal{OT}}$ only the opening information for the IDTCs $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are sent. Therefore once that a valid third round is received, it is possible to replay it in order to answer to many second rounds sent by a malicious sender. Roughly, whether the third round of $\Pi^{\gamma}_{\mathcal{OT}}$ is accepting or not is independent of what a malicious sender sends in the second round. Therefore we have proved that $\Pi^{\gamma}_{\mathcal{OT}}$ has a *replayable* third round. In order to prove that $\Pi^{\gamma}_{\mathcal{OT}}$ is one-sided simulatable secure for $F_{\mathcal{OT}}$ (see Definition 10) we divide the security proof in two parts; the former proves the security against a malicious sender, and the latter proves the security against a malicious receiver. More precisely we prove that $\Pi^{\gamma}_{\mathcal{OT}}$ is secure against a malicious receiver for an arbitrary chosen $\gamma = \mathsf{poly}(\lambda)$, and is secure against malicious sender for $\gamma = 1$ (i.e. when just the first four rounds of the protocol are executed).

Security against a malicious sender. In this case we just need to prove that the output of $S^{\star}_{\mathcal{OT}}$ of the execution of $\Pi^{\gamma}_{\mathcal{OT}}$ when $R_{\mathcal{OT}}$ interacts with $S^{\star}_{\mathcal{OT}}$ using $b = 0$ as input is computationally indistinguishable from when $R_{\mathcal{OT}}$ uses $b = 1$ as input. The differences between these two hybrid experiments consist of the message committed in $\mathtt{com}$ and the way in which the IDTCs are computed. More precisely, in the first experiment, when $b = 0$ is used as input, $\mathtt{tcom}_0$ and the corresponding opening $(\mathtt{tdec}_0^1, r_0^1)$ are computed using the trapdoor procedure (in this case the message committed in $\mathtt{com}$ is 0), while $\mathtt{tcom}_1$ and $(\mathtt{tdec}_1^1, r_1^1)$ are computed using the *honest* procedure. In the second experiment, $\mathtt{tcom}_0$ and the respective opening $(\mathtt{tdec}_0^1, r_0^1)$ are computed using the honest procedure, while $\mathtt{tcom}_1$ and $(\mathtt{tdec}_1^1, r_1^1)$ are computed using the trapdoor procedure of the IDTC scheme. In order to prove the indistinguishability between these two experiments we proceed via hybrid arguments. The first hybrid experiment $\mathcal{H}_1$ is equal to when $R_{\mathcal{OT}}$ interacts with against $S^\star_{\mathcal{OT}}$ according $\Pi^\gamma_{\mathcal{OT}}$ when $b = 0$ is used as input. In $\mathcal{H}_2$ the honest procedure of IDTC is used instead of the trapdoor one in order to compute $\mathtt{tcom}_0$ and the opening $(\mathtt{tdec}_0^1, r_0^1)$. We observe that in $\mathcal{H}_2$ both the IDTCs are computed using the honest procedure, therefore no trapdoor information (i.e. the randomness used to compute $\mathtt{com}$) is required. The computational-indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ comes from the trapdoorness of the IDTC TC_0. In $\mathcal{H}_3$ the value committed in $\mathtt{com}$ goes from 0 to 1. $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable due to the hiding of PBCOM. It is important to observe that a reduction to the hiding of PBCOM is possible because the randomness used to compute $\mathtt{com}$ is no longer used in the protocol execution to run one of the IDTCs. In the last hybrid experiment $\mathcal{H}_4$ the trapdoor procedure is used in order to compute $\mathtt{tcom}_1$ and the opening $(\mathtt{tdec}_1^1, r_1^1)$. We observe that it is possible to run the trapdoor procedure for TC_1 because the message committed in $\mathtt{com}$ is 1. The indistinguishability between $\mathcal{H}_3$ and $\mathcal{H}_4$ comes from the trapdoorness of the IDTC. The observation that $\mathcal{H}_4$ corresponds to the experiment where the honest receiver executes $\Pi^\gamma_{\mathcal{OT}}$ using $b = 1$ as input concludes the security proof.

Security against a malicious receiver. In order to prove that $\Pi^\gamma_{\mathcal{OT}}$ is simulation-based secure against malicious receiver $R^\star_{\mathcal{OT}}$ we need to show a PPT simulator Sim that, having only access to the ideal world functionality $F_{\mathcal{OT}}$, can simulate the output of any malicious $R^\star_{\mathcal{OT}}$ running one execution of $\Pi^\gamma_{\mathcal{OT}}$ with an honest sender $S_{\mathcal{OT}}$. The simulator Sim works as follows. Having oracle access to $R^\star_{\mathcal{OT}}$, Sim runs as a sender in $\Pi^\gamma_{\mathcal{OT}}$ by sending two random strings R_0 and R_1 and the pair of TDPs $f_{0,1}$ and $f_{1,1}$ in the second round. Let $(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)$ be the messages sent in the third round by $R^\star_{\mathcal{OT}}$. Now Sim rewinds $R^\star_{\mathcal{OT}}$ by sending two fresh random strings $\overline{R}_0$ and $\overline{R}_1$ such that $\overline{R}_0 \neq R_0$ and $\overline{R}_1 \neq R_1$.

Let $(\overline{\mathtt{tdec}}_0^1, \overline{r}_0^1), (\overline{\mathtt{tdec}}_1^1, \overline{r}_1^1)$ be the messages sent in the third round by $R^\star_{\mathcal{OT}}$ after this rewind, then there are only two things that can happen[13]:

1. $r^1_{b^\star} \neq \overline{r}^1_{b^\star}$ and $r^1_{1-b^\star} = \overline{r}^1_{1-b^\star}$ for some $b^\star \in \{0, 1\}$ or
2. $r_0^1 = \overline{r}_0^1$ and $r_1^1 = \overline{r}_1^1$.

More precisely, due to the perfect binding of PBCOM at most one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be opened to a different message. Therefore $R^\star_{\mathcal{OT}}$ can either open both $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ to the same messages r_0^1 and r_1^1, or change in the opening of at most one of them. This yields to the following important observation. If one among r_0^1 and r_1^1 changes during the rewind, let us say $r_{b^\star}$ for $b^\star \in \{0, 1\}$ (case 1), then the input bit used by $R^\star_{\mathcal{OT}}$ has to be $b^\star$. Indeed we recall that the only efficient way (i.e. without inverting the TDP) for a receiver to get the output is to equivocate one of the IDTCs in order to compute the inverse of one between $R_0 \oplus r_0^1$ and $R_1 \oplus r_1^1$. Therefore the simulator invokes the ideal world functionality $F_{\mathcal{OT}}$ using $b^\star$ as input, and upon receiving $l_{b^\star}$ computes $W^1_{b^\star} = l_{b^\star} \oplus \mathsf{hc}(f^{-\lambda}_{b^\star,1}(r^1_{b^\star} \oplus R_{b^\star}))$ and sets $W^1_{1-b^\star}$ to a random string. Then sends W_0^1 and

[13] $R^\star_{\mathcal{OT}}$ could also abort after the rewind. In this case we use the following standard argument. If p is the probability of $R^\star_{\mathcal{OT}}$ of giving an accepting third round, λ/p rewinds are made until $R^\star_{\mathcal{OT}}$ gives another answer.

W_1^1 with two freshly generated TDPs $f_{0,2}, f_{1,2}$ (according to the description of $\Pi_{\mathcal{OT}}^{\gamma}$ given in Fig. 5) to $R^\star_{\mathcal{OT}}$. Let us now consider the case where the opening of $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ stay the same after the rewinding procedure (case two). In this case, Sim comes back to the main thread and sets both W_0^1 and W_1^1 to a random string. Intuitively if $R^\star_{\mathcal{OT}}$ does not change neither r_0^1 nor r_1^1 after the rewind, then his behavior is not adaptive on the second round sent by Sim. Therefore, he will be able to compute the inverse of neither $R_0 \oplus r_0^1$ nor $R_1 \oplus r_1^1$. That is, both $R_0 \oplus r_0^1$ and $R_1 \oplus r_1^1$ would be the results of the execution of two coin-flipping protocols, therefore both of them are difficult to invert without knowing the trapdoors of the TDPs. This implies that $R^\star_{\mathcal{OT}}$ has no efficient way to tells apart whether W_0^1 and W_1^1 are random strings or not.

Completed the fourth round, for $i = 2, \dots, \gamma$, Sim continues the interaction with $R^\star_{\mathcal{OT}}$ by always setting both W_0^i and W_1^i to a random string when $r_0^1 = r_0^i$ and $r_1^1 = r_1^i$, and using the following strategy when $r_{b^\star}^1 \neq r_{b^\star}^i$ and $r_{1-b^\star}^1 = r_{1-b^\star}^i$ for some $b^\star \in \{0,1\}$. Sim invokes the ideal world functionality $F_{\mathcal{OT}}$ using $b^\star$ as input, and upon receiving $l_{b^\star}$ computes $W_{b^\star}^i = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,i}^{-\lambda}(r_{b^\star}^i \oplus R_{b^\star}))$, sets $W_{1-b^\star}^i$ to a random string and sends with them two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R^\star_{\mathcal{OT}}$. When the interaction against $R^\star_{\mathcal{OT}}$ is over, Sim stops and outputs what $R^\star_{\mathcal{OT}}$ outputs. We observe that the simulator needs to invoke the ideal world functionality just once. Indeed, we recall that only one of the IDTCs can be equivocated, therefore once that the bit $b^\star$ is decided (using the strategy described before) it cannot change during the simulation. The last thing that remains to observe is that it could happen that Sim never needs to invoke the ideal world functionality in the case that: 1) during the rewind the values (r_0^1, r_1^1) stay the same; 2) $r_b^i = r_b^j$ for all $i, j \in \{1, \dots, \gamma\}$ and all $b = \{0,1\}$. In this case Sim, even though it does not need to query the ideal functionality to internally complete an interaction with $R^\star_{\mathcal{OT}}$ we assume, without loss of generality, that Sim invokes the ideal functionality by using a random bit $b^\star \in \{0,1\}$.

We formally prove that the output of Sim is computationally indistinguishable from the output of $R^\star_{\mathcal{OT}}$ in the real world execution for every $\gamma = \mathsf{poly}(\lambda)$. The proof goes trough hybrid arguments starting from the real world execution. We gradually modify the real world execution until the input of the honest party is not needed anymore such that the final hybrid would represent the simulator for the ideal world. We denote by $\mathsf{OUT}_{\mathcal{H}_i, R^\star_{\mathcal{OT}}(z)}(1^\lambda)$ the output distribution of $R^\star_{\mathcal{OT}}$ in the hybrid experiment $\mathcal{H}_i$.

- $\mathcal{H}_0$ is identical to the real execution. More precisely $\mathcal{H}_0$ runs $R^\star_{\mathcal{OT}}$ using fresh randomness and interacts with him as the honest sender would do on input (l_0, l_1).
- $\mathcal{H}_0^{\mathsf{rew}}$ proceeds according to $\mathcal{H}_0$ with the difference that $R^\star_{\mathcal{OT}}$ is rewound up to the second round by receiving two fresh random strings $\overline{R}_0$ and $\overline{R}_1$. This process is repeated until $R^\star_{\mathcal{OT}}$ completes the third round again (every time using different randomness). More precisely, if $R^\star_{\mathcal{OT}}$ aborts after the rewind then a fresh second round is sent up to λ/p times, where p is the probability of $R^\star_{\mathcal{OT}}$ of completing the third round in $\mathcal{H}_0$. If $p = \mathsf{poly}(\lambda)$ then the expected running time of $\mathcal{H}_0^{\mathsf{rew}}$ is $\mathsf{poly}(\lambda)$ and its output is statistically close to the output of $\mathcal{H}_0$. When the third round is completed the hybrid experiment comes back to the main thread and continues according to $\mathcal{H}_0$
- $\mathcal{H}_1$ proceeds according to $\mathcal{H}_0^{\mathsf{rew}}$ with the difference that after the rewinds executes the following steps. Let r_0^1 and r_1^1 be the messages opened by $R^\star_{\mathcal{OT}}$ in the third round of the main thread and $\overline{r}_0^1$ and $\overline{r}_1^1$ be the messages opened during the rewind. We distinguish two cases that could happen:
  1. $r_0^1 = \overline{r}_0^1$ and $r_1^1 = \overline{r}_1^1$ or
  2. $r_{b^\star}^1 \neq \overline{r}_{b^\star}^1$ and $\overline{r}_{1-b^\star}^1 = r_{1-b^\star}^1$ for some $b^\star \in \{0,1\}$.

In this hybrid we assume that the first case happen with non-negligible probability. After the rewind $\mathcal{H}_1$ goes back to the main thread, and in order to compute the fourth round, picks $W_0^1 \leftarrow \{0,1\}^\lambda$ computes $W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$, $(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends $(W_0^1, W_1^1, f_{0,2}, f_{1,2})$ to $R^\star_{\mathcal{OT}}$. Then the experiment continues according to $\mathcal{H}_0$. Roughly, the difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is that in the latter hybrid experiment W_0^1 is a random string whereas in $\mathcal{H}_1$ $W_0^1 = l_0 \oplus \mathsf{hc}(f_{0,1}^{-\lambda}(r_0^1 \oplus R_0))$.

We now prove that the indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ comes from the security of the hardcore bit function for λ bits hc for the TDP $\mathcal{F}$. More precisely, assuming by contradiction that the outputs of $\mathcal{H}_0$ and $\mathcal{H}_1$ are distinguishable we construct and adversary $\mathcal{A}^{\mathcal{F}}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R^\star_{\mathcal{OT}}$ has non-negligible advantage in distinguishing $\mathcal{H}_0^{\mathsf{rew}}$ from $\mathcal{H}_1$ and consider the randomness ρ used by $R^\star_{\mathcal{OT}}$ and the first round computed by $R^\star_{\mathcal{OT}}$ in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^{\mathcal{F}}$, on input the randomness ρ, the messages r_0^1 and r_1^1 executes the following steps.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $(f, H, f^\lambda(x))$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_0 = r_0^1 \oplus f^\lambda(x)$, pick a random string R_1, compute $(f_{1,1}, f_{1,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, set $f_{0,1} = f$ and sends $R_0, R_1, f_{0,1}, f_{1,1}$ to $R^\star_{\mathcal{OT}}$.
3. Upon receiving $(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)$ compute $W_0^1 = l_0 \oplus H$, $W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$, $(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and send $(W_0^1, W_1^1, f_{0,2}, f_{1,2})$. [14]
4. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_1$ (and $\mathcal{H}_0^{\mathsf{rew}}$) and output what $R^\star_{\mathcal{OT}}$ outputs.

This part of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_0^{\mathsf{rew}}$, otherwise $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_1$.

– $\mathcal{H}_2$ proceeds according to $\mathcal{H}_1$ with the difference that both W_0 and W_1 are set to random strings. Also in this case the indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ comes from the security of the hardcore bit function for λ bits hc for the family $\mathcal{F}$ (the same arguments of the previous security proof can be used to prove the indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_1$).

– $\mathcal{H}_3$ In this hybrid experiment we consider the case where after the rewind, with non-negligible probability, $r_{b^\star}^1 \neq \overline{r}_{b^\star}^1$ and $\overline{r}_{1-b^\star}^1 = r_{1-b^\star}^1$ for some $b^\star \in \{0,1\}$.

In this case, in the main thread the hybrid experiment computes $W_{b^\star}^1 = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,1}^{-\lambda}(r_{b^\star}^1 \oplus R_{b^\star}))$, picks $W_{1-b^\star}^1 \leftarrow \{0,1\}^\star$ sends W_0^1, W_1^1 with two freshly generated TDPs $f_{0,2}, f_{1,2}$. $\mathcal{H}_3$ now continues the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_2$. The indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ comes from the security of the hardcore bit function for λ bits hc for the TDP $\mathcal{F}$. More precisely, assuming by contradiction that $\mathcal{H}_2$ and $\mathcal{H}_3$ are distinguishable, we construct and adversary $\mathcal{A}^{\mathcal{F}}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R^\star_{\mathcal{OT}}$ has non-negligible advantage in distinguish $\mathcal{H}_2$ from $\mathcal{H}_3$ and consider the randomness ρ used by $R^\star_{\mathcal{OT}}$ and the first round computed in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^{\mathcal{F}}$, on input the randomness ρ, the message $b^\star$ committed in $\mathtt{com}$ and the message $r_{1-b^\star}^1$ committed $\mathtt{tcom}_{1-b^\star}$, $\mathcal{A}^{\mathcal{F}}$ executes the following steps.

[14] Observe that $R^\star_{\mathcal{OT}}$ could send values different from r_0^1 and r_1^1 in the third round. In this case $\mathcal{A}^{\mathcal{F}}$ just recomputes the second round using fresh randomness and asking another challenge $\overline{f}, \overline{H}, \overline{f}^\lambda(x)$ to the challenger until in the third round the messages r_0^1 and r_1^1 are received again. This allows $\mathcal{A}^{\mathcal{F}}$ to break the security of $\overline{f}$ because we are assuming that in this experiment $R^\star_{\mathcal{OT}}$ opens, with non-negligible probability, $\mathtt{tcom}_0$ to r_0^1 and $\mathtt{tcom}_1$ to r_1^1.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $(f, H, f^\lambda(x))$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_{1-b^\star} = r^1_{1-b^\star} \oplus f^\lambda(x)$, pick a random string $R_{b^\star}$, computes $(f_{b^\star,1}, f^{-1}_{b^\star,1}) \leftarrow \mathsf{Gen}(1^\lambda)$, sets $f_{1-b^\star,1} = f$ and send $(R_0, R_1, f_{0,1}, f_{1,1})$ to $R^\star_{\mathcal{OT}}$.
3. Upon receiving $(\mathtt{tdec}^1_0, r^1_0), (\mathtt{tdec}^1_1, r^1_1)$ compute $W^1_{1-b^\star} = l_{1-b^\star} \oplus H$, $W^1_{b^\star} = l_{b^\star} \oplus \mathsf{hc}(f^{-\lambda}_{b^\star,1}(r^1_{b^\star} \oplus R_{b^\star}))$, $(f_{0,2}, f^{-1}_{0,2}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f^{-1}_{1,2}) \leftarrow \mathsf{Gen}(1^\lambda)$ and send $(W^1_0, W^1_1, f_{0,2}, f_{1,2})$.
4. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_2$ (and $\mathcal{H}_3$) and output what $R^\star_{\mathcal{OT}}$ outputs.

This part of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_2$, otherwise he acts as in $\mathcal{H}_3$.

– $\mathcal{H}^j_3$ proceeds according to $\mathcal{H}_3$ with the differences that for $i = 2, \dots, j$
 1. if $r^i_{b^\star} \neq r^1_{b^\star}$ for some $b^\star \in \{0,1\}$ then $\mathcal{H}^j_3$ picks $W^i_{1-b^\star} \leftarrow \{0,1\}^\lambda$, computes $W^i_{b^\star} = l_{b^\star} \oplus \mathsf{hc}(f^{-\lambda}_{b^\star,i}(r^i_{b^\star} \oplus R_{b^\star}))$ and sends W^i_0, W^i_i with two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R^\star_{\mathcal{OT}}$ otherwise
 2. $\mathcal{H}^j_3$ picks $W^i_0 \leftarrow \{0,1\}^\lambda$ and $W^i_1 \leftarrow \{0,1\}^\lambda$ and sends W^i_0, W^i_1 with two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R^\star_{\mathcal{OT}}$.

Roughly speaking, if $R^\star_{\mathcal{OT}}$ changes the opened message w.r.t. $\mathtt{tcom}_{b^\star}$, then $W^i_{b^\star}$ is correctly computed and $W^i_{1-b^\star}$ is sets to a random string. Otherwise, if the opening of $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ stay the same as in the third round, then both W^i_0 and W^i_1 are random strings (for $i = 2, \dots, j$). We show that $\mathsf{OUT}_{\mathcal{H}^{j-1}_3, R^\star_{\mathcal{OT}}(z)}(1^\lambda) \approx \mathsf{OUT}_{\mathcal{H}^j_3, R^\star_{\mathcal{OT}}(z)}(1^\lambda)$ in two steps. In the first step we show that the indistinguishability between these two hybrid experiments holds for the first case (when $r^i_{b^\star} \neq r^1_{b^\star}$ for some bit $b^\star$), and in the second step we show that the same holds when $r^i_0 = r^1_0$ and $r^i_1 = r^1_1$.

We first recall that if $r^i_{b^\star} \neq r^1_{b^\star}$, then $\mathtt{tcom}_{1-b^\star}$ is perfectly binding, therefore we have that $r^i_{1-b^\star} = r^1_{1-b^\star}$. Assuming by contradiction that $\mathcal{H}^{j-1}_3$ and $\mathcal{H}^j_3$ are distinguishable then we construct and adversary $\mathcal{A}^{\mathcal{F}}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R^\star_{\mathcal{OT}}$ has non-negligible advantage in distinguishing $\mathcal{H}^{j-1}_3$ from $\mathcal{H}^j_3$ and consider the randomness ρ used by $R^\star_{\mathcal{OT}}$ and the first round computed by $R^\star_{\mathcal{OT}}$ in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^{\mathcal{F}}$, on input the randomness ρ, the message $b^\star$ committed in $\mathtt{com}$ and the message $r^1_{1-b^\star}$ committed $\mathtt{tcom}_{1-b^\star}$, executes the following steps.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $f, H, f^\lambda(x)$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_{1-b^\star} = r^1_{1-b^\star} \oplus f^\lambda(x)$, pick a random string $R_{b^\star}$, compute $(f_{0,1}, f^{-1}_{0,1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $(f_{1,1}, f^{-1}_{1,1}) \leftarrow \mathsf{Gen}(1^\lambda)$ send $R_0, R_1, f_{0,1}, f_{1,1}$ to $R^\star_{\mathcal{OT}}$.
3. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}^{j-1}_3$ using $f_{1-b^\star,j} = f$ instead of using the generation function $\mathsf{Gen}(\cdot)$ when it is required.
4. Upon receiving $(\mathtt{tdec}^j_0, r^j_0), (\mathtt{tdec}^j_1, r^j_1)$ compute $W^j_{1-b^\star} = l_{1-b^\star} \oplus H$,[15] $W^j_{b^\star} = l_{b^\star} \oplus \mathsf{hc}(f^{-\lambda}_{b^\star,j}(r^j_{b^\star} \oplus R_{b^\star}))$, $(f_{0,j+1}, f^{-1}_{0,j+1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,j+1}, f^{-1}_{1,j+1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends $(W^{j+1}_0, W^{j+1}_1, f_{0,j+1}, f_{1,j+1})$.
5. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}^{j-1}_3$ (and $\mathcal{H}^j_3$) and output what $R^\star_{\mathcal{OT}}$ outputs.

[15] It is important to observe that $r^1_{b^\star} = r^j_{b^\star}$.

This step of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^{\star}_{\mathcal{OT}}$ acts as in $\mathcal{H}_3^{j-1}$, otherwise he acts as in $\mathcal{H}_3^{j}$.

The second step of the security proof is almost identical to the proof used to argue the indistinguishability between the outputs of $\mathcal{H}_0$ and $\mathcal{H}_2$.

The entire security proof is almost over, indeed the output of $\mathcal{H}_3^{\gamma}$ corresponds to the output of the simulator Sim and $\mathsf{OUT}_{\mathcal{H}_3,R^{\star}_{\mathcal{OT}}(z)}(1^{\lambda}) = \mathsf{OUT}_{\mathcal{H}_3^1,R^{\star}_{\mathcal{OT}}(z)}(1^{\lambda}) \approx \mathsf{OUT}_{\mathcal{H}_3^2,R^{\star}_{\mathcal{OT}}(z)}(1^{\lambda}) \cdots \approx \mathsf{OUT}_{\mathcal{H}_3^{\gamma},R^{\star}_{\mathcal{OT}}(z)}(1^{\lambda})$. Therefore we can claim that the output of $\mathcal{H}_0$ is indistinguishable from the output of Sim when at most one between l_0 and l_1 is used.

$\square$

Theorem 2. *Assuming TDPs, for any $\gamma > 0$ $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^{m}_{\mathcal{OT}}$ with one-sided simulation. Moreover the third round is* replayable.

Proof. The third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is *replayable* due to the same arguments used in the security proof of Theorem 1. We now prove that $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^{m}_{\mathcal{OT}}$ with one-sided simulation according to Definition 12. More precisely to prove the security against the malicious sender $S^{\star}_{\overrightarrow{\mathcal{OT}}}$ we start by consider the execution $\mathcal{H}_0$ that correspond to the real execution where the input $b_1, \dots, b_m$ is used by the receiver and then we consider the experiment $\mathcal{H}_i$ where the input used by the receiver is $1-b_1, \dots, 1-b_i, b_{i+1}, \dots, b_m$. Suppose now by contradiction that the output distributions of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ (for some $i \in \{1, m-1\}$) are distinguishable, then we can construct a malicious sender $S^{\star}_{\mathcal{OT}}$ that breaks the security of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious sender. This allow us to claim that the output distribution of $\mathcal{H}_0$ is indistinguishable from the output distribution of $\mathcal{H}_m$. A similar proof can be made when the malicious party is the receiver. More precisely, we start by consider the execution $\mathcal{H}_0$ that correspond to the real execution where the input $((l_0^1, l_1^1), \dots, (l_0^m, l_1^m))$ is used by the sender and then we consider the experiment $\mathcal{H}_i$ where the simulator instead of the honest sender procedure is used in the first i parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$. Supposing by contradiction that the output distributions of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ (for some $i \in \{1, m-1\}$) are distinguishable, then we can construct a malicious receiver $R^{\star}_{\mathcal{OT}}$ that breaks the security of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious sender. We observe that in $\mathcal{H}_i$ in the first i parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$ the simulator Sim is used and this could disturb the reduction to the security of $\Pi^{\gamma}_{\mathcal{OT}}$ when proving that the output distribution of $\mathcal{H}_i$ is indistinguishable from the output distribution of $\mathcal{H}_{i+1}$. In order to conclude the security proof we need just to show that Sim's behaviour does not disturb the reduction. As described in the security proof of $\Pi^{\gamma}_{\mathcal{OT}}$, the simulation made by Sim roughly works by rewinding from the third to the second round while from the fourth round onwards Sim works straight line. An important feature enjoyed by Sim is that he maintains the main thread. Let $\mathcal{C}^{\mathcal{OT}}$ be the challenger of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious receiver, our adversary $R^{\star}_{\mathcal{OT}}$ works as following.

1. Upon receiving the first round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$, forward the $(i+1)$-th component ot_1 to $\mathcal{C}^{\mathcal{OT}}$[16].
2. Upon receiving ot_2 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ by computing the second round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_2 is used.
3. Upon receiving the third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$, forward the $(i+1)$-th component ot_3 to $\mathcal{C}^{\mathcal{OT}}$.

[16] We recall that $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is constructed by executing in parallel m instantiations of $\Pi^{\gamma}_{\mathcal{OT}}$, therefore in this reduction we are just replacing the $(i+1)$-th component of every rounds sent to $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ with the value received by $\mathcal{C}^{\mathcal{OT}}$. Vice versa, we forward to $\mathcal{C}^{\star}$ the $(i+1)$-th component of the rounds received from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$.

4. Upon receiving ot_4 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^\star_{\overrightarrow{\mathcal{OT}}}$ by computing the fourth round of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_4 is used.
5. for $i = 2, \dots, \gamma$ follow the strategy described in step 3 and 4 and output what $R^\star_{\overrightarrow{\mathcal{OT}}}$ outputs.

We recall that in $\mathcal{H}_i$ (as well as in $\mathcal{H}_{i+1}$) in the first i execution of $\Pi^\gamma_{\mathcal{OT}}$ the simulator is used, therefore a rewind is made from the third to the second round. During the rewinds $R^\star_{\mathcal{OT}}$ can forward to $R^\star_{\overrightarrow{\mathcal{OT}}}$ the same second round ot_2. Moreover, due to the main thread property enjoyed by Sim, after the rewind $R^\star_{\mathcal{OT}}$ can continue the interaction against $R^\star_{\overrightarrow{\mathcal{OT}}}$ without rewind $\mathcal{C}^\star$. Indeed if Sim does not maintains the main thread then, even though the same ot_2 is used during the rewind, $R^\star_{\overrightarrow{\mathcal{OT}}}$ could send a different ot_3 making impossible to efficiently continue the reduction.

□

4 Secure 2PC in the Simultaneous Message Exchange Model

Overview of our protocol: $\Pi_{2\mathcal{PC}} = (P_1, P_2)$**.** In this section we give an high-level overview of our 4-round 2PC protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$ to compute every functionality $F = (F_1, F_2)$ in the simultaneous message exchange model. $\Pi_{2\mathcal{PC}}$ consists of two simultaneous symmetric executions of the same subprotocol in which only one party learns the output. In the rest of the paper we indicate as left execution the execution of the protocol where P_1 learns the output and as right execution the execution of the protocol where P_2 learns the output. In Fig. 6 we provide the high level description of the left execution of $\Pi_{2\mathcal{PC}}$. We denoted by (m_1, m_2, m_3, m_4) the messages played in the left execution where (m_1, m_3) are sent by P_1 and (m_2, m_4) are sent by P_2. Likewise, in the right execution of the protocol the messages are denoted by $(\tilde{m}_1, \tilde{m}_2, \tilde{m}_3, \tilde{m}_4)$ where $(\tilde{m}_1, \tilde{m}_3)$ are sent by P_2 and $(\tilde{m}_2, \tilde{m}_4)$ are sent by P_1. Therefore, messages $(m_j, \tilde{m}_j)$ are exchanged simultaneously in the j-th round, for $j \in \{1, \dots, 4\}$. Our construction uses the following tools.

- A non-interactive perfectly binding computationally hiding commitment scheme $\mathsf{PBCOM} = (\mathsf{Com}, \mathsf{Dec})$.
- A Yao's garbled circuit scheme $(\mathsf{GenGC}, \mathsf{EvalGC})$ with simulator SimGC.
- The protocol $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$ of Sec. 3 that securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation and has the third round replayable.
- A 4-round delayed-input NMZK AoK $\mathsf{NMZK} = (\mathcal{P}_{\mathsf{NMZK}}, \mathcal{V}_{\mathsf{NMZK}})$ for the $\mathcal{NP}$-language L_{NMZK} that will be specified later (see Sec. 4.1 for the formal definition of L_{NMZK}).

In Figure 6 we propose the high-level description of the left execution of $\Pi_{2\mathcal{PC}}$ where P_1 runs on input $x \in \{0,1\}^\lambda$ and P_2 on input $y \in \{0,1\}^\lambda$.

$P_1(x)$ | | $P_2(y)$
Run $R_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, x$ and randomness α to get ot^1;
Run $\mathcal{V}_{\mathsf{NMZK}}$ on input 1^λ to get nmzk^1.

$m_1 = (\mathsf{ot}^1, \mathsf{nmzk}^1)$ $\longrightarrow$

$(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F_1, y; \omega)$;
$(\mathtt{com_L}, \mathtt{dec_L}) \leftarrow \mathsf{Com}(Z_{1,0}||Z_{1,1}||, \dots, ||Z_{\lambda,1})$;
$(\mathtt{com}_{\mathsf{GC_y}}, \mathtt{dec}_{\mathsf{GC_y}}) \leftarrow \mathsf{Com}(\mathsf{GC_y})$;
Run $S_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, \mathsf{ot}^1$, $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1})$ and randomness β to get ot^2;
Run $\mathcal{P}_{\mathsf{NMZK}}$ on input 1^λ and nmzk^1 to get nmzk^2.

$m_2 = (\mathtt{com_L}, \mathsf{ot}^2, \mathtt{com}_{\mathsf{GC_y}}, \mathsf{nmzk}^2)$ $\longleftarrow$

Run $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^2 to get ot^3;
Run $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^2 to get nmzk^3.

$m_3 = (\mathsf{ot}^3, \mathsf{nmzk}^3)$ $\longrightarrow$

Run $S_{\overrightarrow{\mathcal{OT}}}$ on input ot^3 to get ot^4;
Run $\mathcal{P}_{\mathsf{NMZK}}$ on input nmzk^3, stm[a] and w_{stm}[b] to get nmzk^4.

$m_4 = (\mathsf{ot}^4, \mathsf{GC_y}, \mathsf{nmzk}^4)$ $\longleftarrow$

Run $R_{\overrightarrow{\mathcal{OT}}}$ in input ot^4 thus obtaining $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$;
If $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^4 and stm outputs 1
output $v = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})$;
otherwise output $\perp$.

[a] Informally, NMZK proves that: 1) P_2 has performed both oblivious transfers correctly using the same input y; 2) the Yao's gabled circuit $\mathsf{GC_y}$ is correctly computed and 3) the value $\mathsf{GC_y}$ sent in the last round represents the message committed in $\mathtt{com}_{\mathsf{GC_y}}$.

[b] w_{stm} is s.t. $(\mathsf{stm}, w_{\mathsf{stm}}) \in \mathsf{Rel}_{\mathsf{L_{NMZK}}}$.

Figure 6: High-level description of the left execution of $\Pi_{2\mathcal{PC}}$.

4.1 Formal Description of Our $\Pi_{2\mathcal{PC}} = (P_1, P_2)$

We first start by defining the following $\mathcal{NP}$-language

$$
\begin{aligned}
L_{\mathsf{NMZK}} = \Big\{ \big(\mathsf{com}_{\mathsf{GC}}, \mathsf{com}_{\mathsf{L}}, \mathsf{GC}, (\mathsf{ot}^1, \mathsf{ot}^2, \mathsf{ot}^3, \mathsf{ot}^4)\big) :& \\
\exists(\mathsf{dec}_{\mathsf{GC}}, \mathsf{dec}_{\mathsf{L}}, \mathsf{input}, \alpha, \beta, \omega) \text{ s.t.}& \\
\big((Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC}) \leftarrow \mathsf{GenGC}(1^\lambda, F_1, \mathsf{input}; \omega)\big) \text{ AND}& \\
\big(\mathsf{Dec}(\mathsf{com}_{\mathsf{L}}, \mathsf{dec}_{\mathsf{L}}, Z_{1,0}||Z_{1,1}||, \ldots, ||Z_{\lambda,0}||Z_{\lambda,1}) = 1\big) \text{ AND}& \\
\big(\mathsf{ot}^1 \text{ and } \mathsf{ot}^3 \text{are obtained by running } R_{\overrightarrow{\mathcal{OT}}} \text{ on input } 1^\lambda, \mathsf{input}, \alpha\big) \text{ AND}& \\
\big(\tilde{\mathsf{ot}}^2 \text{ and } \tilde{\mathsf{ot}}^4 \text{ are obtained by running } S_{\overrightarrow{\mathcal{OT}}} \text{ on input}& \\
(1^\lambda, Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \beta)\big)\Big\}.&
\end{aligned}
$$

The NMZK AoK NMZK used in our protocol is for the $\mathcal{NP}$-language L_{NMZK} described above. Now we are ready to describe our protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$ in a formal way.

Protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$

Common input: security parameter λ and instance length ℓ_{NMZK} of the statement of the NMZK.
P_1's input: $x \in \{0,1\}^\lambda$, *P_2's input:* $y \in \{0,1\}^\lambda$.

Round 1. In this round P_1 sends the message m_1 and P_2 the message $\tilde{m}_1$. The steps computed by P_1 to construct m_1 are the following.

1. Run $\mathcal{V}_{\mathsf{NMZK}}$ on input the security parameter 1^λ and ℓ_{NMZK} thus obtaining the first round nmzk^1 of NMZK.
2. Run $R_{\overrightarrow{\mathcal{OT}}}$ on input 1^λ, x and the randomness α thus obtaining the first round ot^1 of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$.
3. Set $m_1 = \big(\mathsf{nmzk}^1, \mathsf{ot}^1\big)$ and send m_1 to P_2.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_1 = \big(\tilde{\mathsf{nmzk}}^1, \tilde{\mathsf{ot}}^1\big)$.

Round 2. In this round P_2 sends the message m_2 and P_1 the message $\tilde{m}_2$. The steps computed by P_2 to construct m_2 are the following.

1. Compute $(Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F_2, y; \omega)$.
2. Compute $(\mathsf{com}_{\mathsf{GC_y}}, \mathsf{dec}_{\mathsf{GC_y}}) \leftarrow \mathsf{Com}(\mathsf{GC_y})$ and
$(\mathsf{com}_{\mathsf{L}}, \mathsf{dec}_{\mathsf{L}}) \leftarrow \mathsf{Com}(Z_{1,0}||Z_{1,1}||, \ldots, ||Z_{\lambda,0}||Z_{\lambda,1})$[17].
3. Run $\mathcal{P}_{\mathsf{NMZK}}$ on input 1^λ and nmzk^1 thus obtaining the second round nmzk^2 of NMZK.
4. Run $S_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}$, ot^1 and the randomness β thus obtaining the second round ot^2 of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$.
5. Set $m_2 = \big(\mathsf{ot}^2, \mathsf{com}_{\mathsf{L}}, \mathsf{com}_{\mathsf{GC_y}}, \mathsf{nmzk}^2\big)$ and send m_2 to P_1.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_2 = \big(\tilde{\mathsf{ot}}^2, \tilde{\mathsf{com}}_{\mathsf{L}}, \tilde{\mathsf{com}}_{\tilde{\mathsf{GC}}_x}, \tilde{\mathsf{nmzk}}^2\big)$.

Round 3. In this round P_1 sends the message m_3 and P_2 the message $\tilde{m}_3$. The steps computed by P_1 to construct m_3 are the following.

1. Run $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^2 thus obtaining the third round nmzk^3 of NMZK.
2. Run $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^2 thus obtaining the third round ot^3 of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$.
3. Set $m_3 = \big(\mathsf{nmzk}^3, \mathsf{ot}^3\big)$ and send m_3 to P_2.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_3 = \big(\tilde{\mathsf{nmzk}}^3, \tilde{\mathsf{ot}}^3\big)$.

[17] Instead of one commitment for each label, P_2 commits to the concatenation of all the labels of the garbled circuit $\mathsf{GC_y}$.

Round 4. In this round P_2 sends the message m_4 and P_1 the message $\tilde{m}_4$. The steps computed by P_2 to construct m_4 are the following.

1. Run $S_{\overrightarrow{\mathcal{OT}}}$ on input ot^3, thus obtaining the fourth round ot^4 of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.
2. Set $\mathtt{stm} = (\mathtt{com}_{\mathsf{GC_y}}, \mathtt{com}_{\mathsf{L}}, \mathsf{GC_y}, \tilde{\mathsf{ot}}_1, \mathsf{ot}_2, \tilde{\mathsf{ot}}_3, \mathsf{ot}_4)$ and $w_{\mathtt{stm}} = (\mathtt{dec}_{\mathsf{GC_y}}, \mathtt{dec}_{\mathsf{L}}, y, \tilde{\alpha}, \beta, \omega)$.
3. Run $\mathcal{P}_{\mathsf{NMZK}}$ on input nmzk^3, $\mathtt{stm}$ and $w_{\mathtt{stm}}$ thus obtaining the fourth round nmzk^4 of NMZK.
4. Set $m_4 = \left(\mathsf{nmzk}^4, \mathsf{ot}^4, \mathsf{GC_y}\right)$ and send m_4 to P_1.

Likewise, P_1 performs the same actions of P_2 constructing message $\tilde{m}_4 = \left(\tilde{\mathsf{nmzk}}^4, \tilde{\mathsf{ot}}^4, \tilde{\mathsf{GC}}_x\right)$.

Output computation.

P_1**'s output:** P_1 checks if the transcript $(\mathsf{nmzk}^1, \mathsf{nmzk}^2, \mathsf{nmzk}^3, \mathsf{nmzk}^4)$ is accepting w.r.t. $\mathtt{stm}$. In the negative case P_1 outputs $\perp$, otherwise P_1 runs $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^4 thus obtaining $Z_{1,x_1}, \ldots, Z_{\lambda,x_\lambda}$ and computes the output $v_1 = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \ldots, Z_{\lambda,x_\lambda})$.

P_2**'s output:** P_2 checks if the transcript $\tilde{\mathsf{nmzk}}^1, \tilde{\mathsf{nmzk}}^2, \tilde{\mathsf{nmzk}}^3, \tilde{\mathsf{nmzk}}^4$ is accepting w.r.t. $\tilde{\mathtt{stm}}$. In the negative case P_2 outputs $\perp$, otherwise P_2 runs $R_{\overrightarrow{\mathcal{OT}}}$ on input $\tilde{\mathsf{ot}}^4$ thus obtaining $\tilde{Z}_{1,y_1}, \ldots, \tilde{Z}_{\lambda,y_\lambda}$ and computes the output $v_2 = \mathsf{EvalGC}(\tilde{\mathsf{GC}}_x, \tilde{Z}_{1,y_1}, \ldots, \tilde{Z}_{\lambda,y_\lambda})$.

High-level overview of the security proof. Due to the symmetrical nature of the protocol, it is sufficient to prove the security against one party (let this party be P_2). We start with the description of the simulator Sim. Sim starts the simulator of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ $\mathsf{Sim}_{\mathcal{OT}}$ which outputs the input $y^\star$ used by the malicious party. Sim sends $y^\star$ to the ideal functionality F and receives back $(v_1, v_2) = \left(F_1(x, y^\star), F_2(x, y^\star)\right)$. Then, Sim computes $(\tilde{\mathsf{GC}}_\star, (\tilde{Z}_1, \ldots, \tilde{Z}_\lambda)) \leftarrow \mathsf{SimGC}(1^\lambda, F_2, y^\star, v_2)$ and answer to $\mathsf{Sim}_{\mathcal{OT}}$ using $(\tilde{Z}_1, \ldots, \tilde{Z}_\lambda)$ (we recall that Sim acts as the ideal functionality $F^m_{\mathcal{OT}}$ for $\mathsf{Sim}_{\mathcal{OT}}$). Moreover, instead of committing to the labels of Yao's garbled circuit, in the second round Sim commits to 0 and $\tilde{\mathsf{GC}}_\star$ is sent in the last round. Then Sim runs the simulator $\mathsf{Sim}_{\mathsf{NMZK}}$ of NMZK. For the messages of $\Pi_{\mathcal{OT}}$ where P_1 acts as the receiver, Sim runs $R_{\overrightarrow{\mathcal{OT}}}$ on input 0^λ instead of using x. In our security proof we proceed through a sequence of hybrid experiments, where the first one corresponds to the real-world execution and the final represents the execution of Sim in the ideal world. The core idea of our approach is to run the simulator of NMZK, while extracting the input from $P_2^\star$. By running the simulator of NMZK we are able to guarantee that the value extracted via $\mathsf{Sim}_{\mathcal{OT}}$ is correct, even though $P_2^\star$ is receiving proofs for a false statement. Indeed in each intermediate hybrid experiment that we will consider, also the extractor of NMZK is run in order to extract the witness for the theorem proved by $P_2^\star$. In this way we can prove that the value extracted via $\mathsf{Sim}_{\mathcal{OT}}$ is consistent with the input that $P_2^\star$ is using in the other part of the protocol (e.g. in the construction of the garbled circuit and in the execution of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ in which the adversary acts as a sender). For what we have discussed, the simulator of NMZK rewinds first from the third to the second round (to extract the trapdoor), and then from the fourth to the third round (to extract the witness for the statement proved by $P_2^\star$). We need to show that these rewinding procedures do not disturb the security proof when we rely on the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. This is roughly the reason why we require the third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to be replayable and the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against malicious receiver to hold for any $\gamma = \mathsf{poly}(\lambda)$.

Theorem 3. *Assuming TDPs, $\Pi_{2\mathcal{PC}}$ securely computes every two-party functionality $F = (F_1, F_2)$ with black-box simulation.*

Proof. In order to prove that $\Pi_{2\mathcal{PC}}$ securely computes $F = (F_1, F_2)$, we first observe that, due to the symmetrical nature of the protocol, it is sufficient to prove the security against one party (let this party be P_2). We now show that for every adversary $P_2^\star$, there exists an ideal-world adversary (simulator) Sim such that for all inputs x, y of equal length and security parameter λ:

$$\{\mathsf{REAL}_{\Pi_{2\mathcal{PC}},P_2^\star(z)}(1^\lambda, x, y)\} \approx \{\mathsf{IDEAL}_{F,\mathsf{Sim}(z)}(1^\lambda, x, y)\}.$$

Our simulator Sim is the one showed in Sec. 4.1.

In our security proof we proceed through a series of hybrid experiments, where the first one corresponds to the execution of $\Pi_{2\mathcal{PC}}$ between P_1 and $P_2^\star$ (real-world execution). Then, we gradually modify this hybrid experiment until the input of the honest party is not needed anymore, such that the final hybrid would represent the simulator (simulated execution).

We now give the descriptions of the hybrid experiments and of the corresponding security reductions. We denote the output of $P_2^\star$ and the output of the procedure that interacts against $P_2^\star$ on the behalf of P_1 in the hybrid experiment $\mathcal{H}_i$ with $\{\mathsf{OUT}_{\mathcal{H}_i,P_2^\star(z)}(1^\lambda, x, y)\}_{x\in\{0,1\}^\lambda, y\in\{0,1\}^\lambda}$.

- $\mathcal{H}_0$ corresponds to the real executions. More in details, $\mathcal{H}_0$ runs $P_2^\star$ with a fresh randomness, and interacts with it as the honest player P_1 does using x as input. The output of the experiment is $P_2^\star$'s view and the output of P_1. Note that we are guarantee from the soundness of NMZK that $\mathsf{stm} \in L_{\mathsf{NMZK}}$, that is: 1)$P_2^\star$ uses the same input $y^\star$ in both the OT executions; 2) the garbled circuit committed in $\mathtt{com}_{\mathsf{GC_y}}$ and the corresponding labels committed in $\mathtt{com}_{\mathsf{L}}$ are computed using the input $y^\star$; 3) the garbled circuit sent in the last round is actually the one committed in $\mathtt{com}_{\mathsf{GC_y}}$.
- $\mathcal{H}_1$ proceeds in the same way of $\mathcal{H}_0$ except that the simulator $\mathsf{Sim}_{\mathsf{NMZK}}$ of NMZK is used in order to compute the messages of NMZK played by P_1. Note that $\mathsf{Sim}_{\mathsf{NMZK}}$ rewinds $P_2^\star$ from the 3rd to the 2nd round in oder to extract the trapdoor. The indistinguishability between the output distribution of these two hybrids experiments holds from the property 1 of NMZK (see Definition 8). In this, and also in the next hybrids, we prove that Prob $[\ \mathsf{stm} \notin L_{\mathsf{NMZK}}\ ] \leq \nu(\lambda)$. That is, we prove that $P_2^\star$ behaves honestly across the hybrid experiments even though he is receiving a simulated proof w.r.t. NMZK and $\tilde{\mathsf{stm}}$ does not belong to L_{NMZK}. In this hybrid experiment we can prove that if by contradiction this probability is non-negligible, then we can construct a reduction that breaks the property 2 of NMZK (see Definition 8). Indeed, in this hybrid experiment, the theorem that $P_2^\star$ receives belongs to L_{NMZK} and the simulator of $\mathsf{Sim}_{\mathsf{NMZK}}$ is used in order to compute and accepting transcript w.r.t. NMZK. Therefore, relying on property 2 of Definition 8 we know that there exists a simulator that extracts the the witness for the statement stm proved by $P_2^\star$ with all but negligible probability.
- $\mathcal{H}_2$ proceeds in the same way of $\mathcal{H}_1$ except that the simulator of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$, $\mathsf{Sim}_{\mathcal{OT}}$, is used instead of the sender algorithm $S_{\overrightarrow{\mathcal{OT}}}$. We recall that $\mathsf{Sim}_{\mathcal{OT}}$ requires to interact with the ideal functionality $F^m_{\mathcal{OT}}$. In this case the hybrid experiment $\mathcal{H}_2$ acts on the behalf of the $F^m_{\mathcal{OT}}$ by answering to a query $y^\star$ made by $\mathsf{Sim}_{\mathcal{OT}}$ using the garbled circuit labels $(\tilde{Z}_{1,y_1^\star}, \dots, \tilde{Z}_{\lambda,y_\lambda^\star})$. From the simulatable security against malicious receiver of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ for every $\gamma = \mathsf{poly}(\lambda)$ follows that the output distributions of $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable. Suppose by contradiction this claim does not hold, then we can show a malicious receiver $R^\star_{\overrightarrow{\mathcal{OT}}}$ that breaks the simulatable security of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ against a malicious receiver. In more details, let $\mathcal{C}_{\mathcal{OT}}$ be the challenger of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$. $R^\star_{\overrightarrow{\mathcal{OT}}}$ plays all the messages of $\Pi_{2\mathcal{PC}}$ as in $\mathcal{H}_1$ ($\mathcal{H}_2$) except for the messages of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$. For these messages $R^\star_{\overrightarrow{\mathcal{OT}}}$ acts as a proxy between $\mathcal{C}_{\mathcal{OT}}$ and $P_2^\star$. In the end of the execution $R^\star_{\overrightarrow{\mathcal{OT}}}$ runs the distinguisher D that distinguishes $\{\mathsf{OUT}_{\mathcal{H}_1,P_2^\star(z)}(1^\lambda, x, y)\}$ from $\{\mathsf{OUT}_{\mathcal{H}_2,P_2^\star(z)}(1^\lambda, x, y)\}$ and outputs what D outputs. We observe that if $\mathcal{C}_{\mathcal{OT}}$ acts as the simulator then $P_2^\star$ acts as

in $\mathcal{H}_2$ otherwise he acts as in $\mathcal{H}_1$.

To argue that $\text{Prob}[\,\mathsf{stm} \notin L_{\mathsf{NMZK}}\,] \leq \nu(\lambda)$ also in this hybrid experiment we use the simulator-extractor $\mathsf{Sim}_{\mathsf{NMZK}}$ in order to check whether the theorem proved by $P_2^\star$ is still true. If it is not the case then we can construct a reduction to the simulatable security against malicious receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. Note that $\mathsf{Sim}_{\mathsf{NMZK}}$ rewinds from the 4th to the 3rd round in order to extract the witness $w_{\mathtt{stm}}$ for the statement $\mathtt{stm}$ proved by $P_2^\star$. These rewinds could cause $P_2^\star$ to ask multiple third rounds of OT $\tilde{\mathsf{ot}}^3_i$ $(i = 1, \ldots, \mathsf{poly}(\lambda))$. In this case $R^\star_{\overrightarrow{\mathcal{OT}}}$ can simply forward $\tilde{\mathsf{ot}}^3_i$ to $\mathcal{C}_{\mathcal{OT}}$ and obtains from $\mathcal{C}_{\mathcal{OT}}$ an additional $\tilde{\mathsf{ot}}^4_i$. This behaviour of $R^\star_{\overrightarrow{\mathcal{OT}}}$ is allowed because $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is simulatable secure against a malicious receiver even when the last two rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are executed γ times (as stated in Theorem 1). Therefore the reduction still works if we set γ equals to the expected number of rewinds that $\mathsf{Sim}_{\mathsf{NMZK}}$ could do. We observe that since we have proved that $\mathsf{stm} \in L_{\mathsf{NMZK}}$, then the value $y^\star$ queried by $\mathsf{Sim}_{\mathcal{OT}}$ corresponds to the input used by $P_2^\star$ in the overall execution of the protocol. That is, $P_2^\star$ uses $y^\star$ to both compute the garbled circuit and to complete the execution $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ in which she acts as a sender. On top of this observation, we obtain that in $\mathcal{H}_1$ the value $v_1 = F_1(x, y^\star)$ corresponds, with overwhelming probability, to the valued computed by running the garbled circuit GC_{y} received by $P_2^\star$ using as input the labels $Z_{1,x_1}, \ldots, Z_{\lambda,x_\lambda}$.

- $\mathcal{H}_3$ differs from $\mathcal{H}_2$ in the way the rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$, where $P_2^\star$ acts as sender, are computed. More precisely instead of using x as input, 0^λ is used. Note that from this hybrid onward it is not possible anymore to compute the output by running EvalGC as in the previous hybrid experiments. This is because we are not able to recover the correct labels to evaluate the garbled circuit. Therefore $\mathcal{H}_3$ computes the output by directly evaluating $v_1 = F_1(x, y^\star)$, where $y^\star$ is the input of $P_2^\star$ obtained by $\mathsf{Sim}_{\mathcal{OT}}$.

  The indistinguishability between the output distributions of $\mathcal{H}_3$ and $\mathcal{H}_2$ comes from the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against malicious sender. Indeed, suppose by contradiction that it is not the case, then we can show a malicious sender $S^\star_{\overrightarrow{\mathcal{OT}}}$ that breaks the indistinguishability based security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against a malicious sender. In more details, let $\mathcal{C}_{\mathcal{OT}}$ be the challenger. $S^\star_{\overrightarrow{\mathcal{OT}}}$ plays all the messages of $\Pi_{2\mathcal{PC}}$ as in $\mathcal{H}_3$ ($\mathcal{H}_2$) except for the messages of OT where he acts as a receiver. For these messages $S^\star_{\overrightarrow{\mathcal{OT}}}$ plays as a proxy between $\mathcal{C}_{\mathcal{OT}}$ and $P_2^\star$. At the end of the execution $S^\star_{\overrightarrow{\mathcal{OT}}}$ runs the distinguisher D that distinguishes $\{\mathsf{OUT}_{\mathcal{H}_2,P_2^\star(z)}(1^\lambda, x, y)\}$ from $\{\mathsf{OUT}_{\mathcal{H}_3,P_2^\star(z)}(1^\lambda, x, y)\}$ and outputs what D outputs. We observe that if $\mathcal{C}_{\mathcal{OT}}$ computes the messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ using the input 0^λ then $P_2^\star$ acts as in $\mathcal{H}_3$ otherwise he acts as in $\mathcal{H}_2$. In this security proof there is another subtlety. During the reduction $S^\star_{\overrightarrow{\mathcal{OT}}}$ runs $\mathsf{Sim}_{\mathsf{NMZK}}$ that rewinds from the third to the second round. This means that $P_2^\star$ could send multiple different second rounds ot^2_i of OT (with $i = 1, \ldots, \mathsf{poly}(\lambda)$). $S^\star_{\overrightarrow{\mathcal{OT}}}$ cannot forward these other messages to $\mathcal{C}_{\mathcal{OT}}$ (he cannot rewind the challenger). This is not a problem because the third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is replayable (as proved in Theorem 1). That is the round ot^3 received from the challenger can be used to answer to any ot^2. To prove that $\text{Prob}[\,\mathsf{stm} \notin L_{\mathsf{NMZK}}\,] \leq \nu(\lambda)$ we use the same arguments as before by observing the the rewinds made by the simulator-extractor from the fourth round to the third one do not affect the reduction.

- $\mathcal{H}_4$ proceeds in the same way of $\mathcal{H}_3$ except for the message committed in $\tilde{\mathtt{com}}_{\mathsf{L}}$. More precisely, instead of computing a commitment of the labels $(\tilde{Z}_{1,0}, \tilde{Z}_{1,1}, \ldots, \tilde{Z}_{\lambda,0}, \tilde{Z}_{\lambda,1})$, a commitment of

$0^\lambda||\ldots||0^\lambda$ is computed. The indistinguishability between the output distributions of $\mathcal{H}_3$ and $\mathcal{H}_4$ follows from the hiding of PBCOM. Moreover, Prob $[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,] \leq \nu(\lambda)$ in this hybrid experiment due to the same arguments used previously.

- $\mathcal{H}_5$ proceeds in the same way of $\mathcal{H}_4$ except for the message committed in $\tilde{\mathtt{com}}_{\mathsf{GC_y}}$: instead of computing a commitment of the Yao's garbled circuit $\tilde{\mathtt{GC}}_x$, a commitment of 0 is computed. The indistinguishability between the output distributions of $\mathcal{H}_4$ and $\mathcal{H}_5$ follow from the hiding of PBCOM. We have that Prob $[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,] \leq \nu(\lambda)$ in this hybrid experiment due to the same arguments used previously.
- $\mathcal{H}_6$ proceeds in the same way of $\mathcal{H}_5$ except that the simulator SimGC is run (instead of GenGC) in order to obtain the Yao's garbled circuit and the corresponding labels. In more details, once $y^\star$ is obtained by $\mathsf{Sim}_{\mathcal{OT}}$ (in the third round), the ideal functionality F is invoked on input $y^\star$. Upon receiving $(v_1, v_2) = \big(F_1(x, y^\star), F_2(x, y^\star)\big)$ the hybrid experiment compute $(\tilde{\mathtt{GC}}_\star, \tilde{Z}_1, \ldots, \tilde{Z}_\lambda) \leftarrow \mathsf{SimGC}(1^\lambda, F_2, y^\star, v_2)$ and replies to the query made by $\mathsf{Sim}_{\mathcal{OT}}$ with $(\tilde{Z}_1, \ldots, \tilde{Z}_\lambda)$. Furthermore, in the 4th round the simulated Yao's garbled circuit $\tilde{\mathtt{GC}}_\star$ is sent, instead of the one generated using GenGC. The indistinguishability between the output distributions of $\mathcal{H}_5$ and $\mathcal{H}_6$ follows from the security of the Yao's garbled circuit. To prove that Prob $[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,] \leq \nu(\lambda)$ we use the same arguments as before by observing the the rewinds made by the simulator-extractor from the fourth round to the third one do not affect the reduction.

The proof ends with the observation that $\mathcal{H}_6$ corresponds to the simulated execution with the simulator Sim.

□

5 Acknowledgments

We thank Ivan Damgård and Claudio Orlandi for remarkable discussions on two-party computations and the suggestion of use public key encryption schemes with special properties instead of trapdoor permutations to construct our oblivious transfer protocol. Research supported in part by "GNCS - INdAM", EU COST Action IC1306, NSF grant 1619348, DARPA, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government.

References

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 468–499, 2017.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Pro-*

ceedings, Part III, volume 9816 of *Lecture Notes in Computer Science*, pages 270–299. Springer, 2016. Full version http://eprint.iacr.org/2016/566.

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 711–742. Springer, 2017. Full version http://eprint.iacr.org/2017/931.

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 127–157, 2017. Full version http://eprint.iacr.org/2016/621.

[CPS+16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2016. Full version http://eprint.iacr.org/2015/810.

[CPS+16b] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92. Springer, 2016. Full version http://eprint.iacr.org/2016/175.

[Dam10] Ivan Damgård. On Σ-protocol. http://www.cs.au.dk/~ivan/Sigma.pdf, 2010.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 205–210. Plenum Press, New York, 1982.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 325–335, 2000.

[GMPP16a] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. Personal communication, August 2016.

[GMPP16b] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien

Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 448–476. Springer, 2016.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.

[Lin10] Yehuda Lindell. Foundations of cryptography 89-856. `http://u.cs.biu.ac.il/~lindell/89-856/complete-89-856.pdf`, 2010.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2015.

[Pol16] Antigoni Polychroniadou. *On the Communication and Round Complexity of Secure Computation*. PhD thesis, Aarhus University, December 2016.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 57–74, 2008.

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer, 2001.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.