Oblivious Transfer from Trapdoor Permutations in Minimal Rounds

Arka Rai Choudhuri[1][0000-0003-0452-3426], Michele Ciampi[2][0000-0001-5062-0388], Vipul Goyal[3], Abhishek Jain[1], and Rafail Ostrovsky[4]

[1] Johns Hopkins University
{achoud,abhishek}@cs.jhu.edu
[2] The University of Edinburgh
michele.ciampi@ed.ac.uk
[3] Carnegie Mellon University and NTT Research
goyal@cs.cmu.edu
[4] University of California, Los Angeles
rafail@cs.ucla.edu

Abstract. Oblivious transfer (OT) is a foundational primitive within cryptography owing to its connection with secure computation. One of the oldest constructions of oblivious transfer was from *certified* trapdoor permutations (TDPs). However several decades later, we do not know if a similar construction can be obtained from TDPs in general.
In this work, we study the problem of constructing round optimal oblivious transfer from trapdoor permutations. In particular, we obtain the following new results (in the plain model) relying on TDPs in a *black-box* manner:

– Three-round oblivious transfer protocol that guarantees indistinguishability-security against malicious senders (and semi-honest receivers).
– Four-round oblivious transfer protocol secure against malicious adversaries with black-box simulation-based security.

By combining our second result with an already known compiler we obtain the first round-optimal 2-party computation protocol that relies in a black-box way on TDPs.
A key technical tool underlying our results is a new primitive we call dual witness encryption (DWE) that may be of independent interest.

Keywords: Two-Party Computation · Trapdoor Permutations · Oblivious Transfer.

1 Introduction

Oblivious transfer (OT) is one of the most recognizable protocols in cryptography. It is a protocol executed by two parties, designated as sender and receiver, with inputs (l_0, l_1) and b respectively. The goal of the protocol is for the receiver to learn l_b, while not learning anything about l_{1-b}. At the same time, the sender should be oblivious to the receiver's input b. The importance of OT is underlined by its fundamental role in cryptography, as it is known to be both necessary and sufficient for secure multiparty computation (MPC) [31]. In fact, recent works [3, 9] further strengthen this connection to devise round-preserving transformations from OT to MPC.

In this work, we revisit the well-studied problem of building *round-optimal* OT in the plain model that are secure against malicious adversaries, who may arbitrarily deviate from the protocol specification. We focus on the task of building such protocols from general assumptions, and in particular, *trapdoor permutations* (TDPs). Roughly speaking, TDPs are permutations that are easy to compute, but hard to invert unless one knows a "trapdoor" (in which case inversion becomes easy).

OT and TDPs are, in fact, historically linked — the first constructions of semi-honest[5] 1-out-of-2 OT protocols [13] were based on TDPs. Subsequent works devised compilation strategies to transform the protocol of [13] to the setting of malicious senders and receivers. In particular, [30] constructed a four-round OT protocol that makes non-black-box use of TDPs. More recently, [35] improved this result by only making *black-box* use of TDPs. Moreover, the round complexity of these protocols is *optimal* (w.r.t. black-box simulation) [30].

A significant disadvantage of these works (including [13]), however, is that when it comes to proving security against malicious adversaries, they require the TDPs to be *certifiable*. Namely, it must be possible to publicly recognize whether a given (possibly adversarially chosen) function is a permutation.

Investigating how to construct complex cryptographic protocols relying on trapdoor permutations is interesting from both the theoretical and the practical perspective.

Indeed, for this reason, the issue of certifiability of TDPs has garnered much interest in the context of the other popular application of TDPs, which is to build non-interactive zero-knowledge (NIZK) [18, 14, 1, 19, 20, 21, 23, 6, 29]. In a similar vein, in this work we ask whether it is possible to forego the reliance on certifiability in building round-optimal OT from TDPs:

Does there exist fully black-box round-optimal OT from trapdoor permutations?

Indeed, one simple way to relax the certifiability requirement is to let the party choosing the TDP proving in zero-knowledge that the TDP was sampled honestly. However this necessarily increases the number of rounds (or requires trusted assumptions). Such an approach has been used in [36], in which the authors show that one-way permutations (without trapdoors) are sufficient to construct OT if one of the two parties is all-powerful. Thus, the problem becomes interesting if one considers the round complexity of constructions.

On the use of Certifiability. To the best of our knowledge, we are not aware of any maliciously secure round-optimal OT protocol that uses the underlying trapdoor permutations even in a *non-black-box* way.

[5] A semi-honest adversary, unlike a malicious adversary, follows the protocol specification. However, it may still try to glean additional information from the execution of the protocol.

In both of the classical applications of TDPs, namely, NIZK and OT, the certifiability property is crucially used for security. In the case of NIZKs, it is used to guarantee soundness against malicious provers in the classical protocol of [14]. In the case of OT, it is used to guarantee security against malicious senders. In both of these applications, one of the parties (the prover, in the case of NIZKs, and the sender, in the case of OT) is required to sample a function f from a family of trapdoor permutations. This is done by sampling an index I via the index generation algorithm of the family of functions. If the party does not sample the index I honestly, the resultant function is no longer guaranteed to be a permutation. In such a scenario, in both of these applications, security completely breaks down (we will give an example hereafter in the paper). A cheating prover is able to break soundness, and a cheating sender is able to break receiver input privacy.

In the context of NIZKs, [1] proposed a technique to address this issue when the TDP family is *full domain*. Here, we say that a TDP family is full domain if the domain is $\{0,1\}^{\mathsf{p}(n)}$ for some polynomial p, else we say that the domain is *partial*. Subsequent works [20, 21, 19, 23] showed that for the case of partial domain, it suffices for one to start with TDPs that are *doubly-enhanced*, i.e., TDPs that additionally have domain and range samplers with additional security properties (see Section 3.1). [6] was able to further relax the requirements for partial domain to only require TDPs that are *public-domain*, i.e. the domain is both efficiently recognizable, and almost uniformly sampleable. In [18] the authors propose a non-interactive proof to certify that the RSA public key specifies a permutation in the random-oracle (RO) model.

These solutions, however, are in the common random string (CRS) model (or in the RO model), and are *not* applicable to our plain model setting. The main technical focus of our work is to eliminate the use of certifiability in building OT, without relying on a CRS or on the RO, and requiring the least possible number of rounds. To achieve this goal, we rely on new notion of *dual witness encryption* (DWE).

1.1 Our Results

We resolve the aforementioned question in the affirmative, and provide details for our result below.

Dual Witness Encryption. As a stepping stone to our solution, we define the notion of dual witness encryption for the pair of disjoint languages (L_0, L_1) such that L_1 is in NP. Intuitively, the notion defines a public-key encryption scheme where the public key (the instance) can either come from L_0, L_1 or may even lie outside the union of these two sets. The scheme guarantees: (i) information theoretic security when encryption is performed using a public-key belonging to the set L_0; and (ii) efficient decryption when encrypted using a public-key belonging to the set L_1 if the decryptor is additionally in possession of a *witness* attesting to this fact.

For use in our OT protocols, we construct a dual witness encryption (DWE) scheme where the public keys will correspond to functions f. Specifically, we build a DWE scheme for (L_0, L_1) where (i) L_0 is the set functions for which a large fraction of points in the domain result in collisions (the reader can think of this as meaning that at least half the points in the domain result in collision on application of functions f in L_0); whereas (ii) L_1 is the set of TDPs output by an honest TDP generation algorithm Gen. While we discuss the details of the *encryption scheme* in the technical overview, for the purposes of this discussion it is helpful to think of an (overly) simplified version of a ciphertext in the encryption scheme to be $(f(k), k \oplus m)$[6] where k is a randomly sampled *key*, and m is the message to be encrypted. Intuitively, if the instance f used to compute an encryption is a function for which many points in the domain have the same image, then $f(k)$ (which is a part of the ciphertext) information theoretically hides the specific key k chosen for encryption, and thereby hides the message m. Instead, if the function f used for the decryption is a TDP, and the randomness used to generate such a function is known, then there exists an efficient procedure that inverts $f(k)$ and decrypts the message. We note that in this case there are instances that belong neither to L_0 nor to L_1 (e.g., the functions for which only a small fraction of points in the domain result in collisions). This is our main tool for tackling uncertifiability. As stated above, this is an oversimplification of our scheme, and we provide more details both for the construction of the tool, and how it is used, in the next section.

As an additional contribution, we show the existence of a dual witness encryption schemes for other languages. For instance the pair of languages (L_0, L_1), where L_0 represents the language of Diffie-Hellman (DH) tuples, and L_1 represents the language of non-DH tuples. In this case, when an encryption is computed using a DH tuple, the encrypted message is information theoretically hidden. In any other case, when the encryption is computed using a tuple that is not DH, it is possible to efficiently decrypt the message. Moreover, the decryption is efficient if the exponents of the non-DH-tuple are known by the decryptor. We also argue that it is possible to extend the above construction to the language of non-Quadratic Residuosity tuples [24][7].

Comparison with similar notions. Dual witness encryption is similar to witness encryption with some important differences: First, we require semantic security to hold even against unbounded adversaries when the instance used for the encryption belongs to L_0. Second, unlike witness encryption, we do not define completeness or hiding for instances that are outside L_0 and L_1.

The notion of *instance-dependent commitment (ID commitment)* [7] enables a committer to commit to a message with respect to an NP language L. When the statement used to compute the commitment is not in L, then the commitment

[6] Note that this is *not* an accurate description of the encryption scheme, but is helpful to provide an intuition.

[7] We note that in this example $L_0 \cup L_1 = \{0,1\}^\star$, but this is not always the case, as we show hereafter.

is statistically hiding, in any other case the commitment is statistically binding. The notion of *extractable ID commitment*, in addition, admits an efficient extraction procedure that on input a commitment computed with respect to an instance in L, outputs the committed message. In [17] the authors show how to construct such an extractable ID commitment scheme for all the languages that admit hash proof systems (e.g., QNR, QR, DDH, DCR). It is easy to see that an extractable ID commitment for the language L is a DWE for the languages (L_0, L_1) with $L_0 = \{0,1\}^\star - L$ and $L_1 = L$. Moreover, any DWE such that $L_0 \cup L_1 = \{0,1\}^\star$ is an extractable ID commitment for the language L_1. The main difference between DWEs and extractable ID commitments is that the extractable ID commitments are defined with respect to one NP-language, whereas our notion provides different guarantees depending on whether the statement is in L_0, L_1 or in neither of the two languages.

Round Optimal Oblivious Transfer. Using Dual Witness Encryption (DWE), we obtain the following results.

Theorem 1 (informal). *Assuming full domain trapdoor permutations, we construct a fully black-box three-round oblivious transfer protocol that is secure against semi-honest receivers and malicious senders.*

Theorem 2 (informal). *Assuming full domain trapdoor permutations, we construct a fully black-box four-round fully simulatable oblivious transfer protocol.*

Round Optimal Two-Party Computation. An immediate corollary from the Theorem 1, in conjunction with the work of [28] building a non-interactive secure two-party protocol in the OT-hybrid model is the following.

Corollary 1. *Assuming full domain trapdoor permutations, there exists a fully black-box round optimal secure two-party computation protocol.*

Functions with partial domain. To the best of our knowledge, to extend the results of previous works [35, 30] in the case of functions with partial domain requires, in addition to the certifiability property, (i) the existence of a sampler which uniformly samples elements from the domain/range; and (ii) the existence of an efficient algorithm that checks whether a given element belongs inside or outside the domain of the function. These properties are called respectively *efficiently sampleable domain/range* and *efficiently recognizable domain.* We show how to extend our theorems and corollary to the case of functions with partial domain by removing the requirement on the function to be certifiable, while maintaining the same requirements of efficiently sampleable domain/range and efficiently recognizable domain.

2 Technical Overview

To illustrate the main ideas underlying this work, it will suffice to assume full domain TDPs, and the extension to partial domains are deferred to the technical sections.

Background: 3-round semi-honest OT. Before we describe the main ideas in our construction, let us recall the basic three-round construction based on enhanced trapdoor permutations (TDPs) in the semi-honest setting (EGL) [13, 30].

Let $l_0, l_1 \in \{0,1\}$ be the input of the sender S and b be the input bit of the receiver R, the construction is presented in Figure 1.

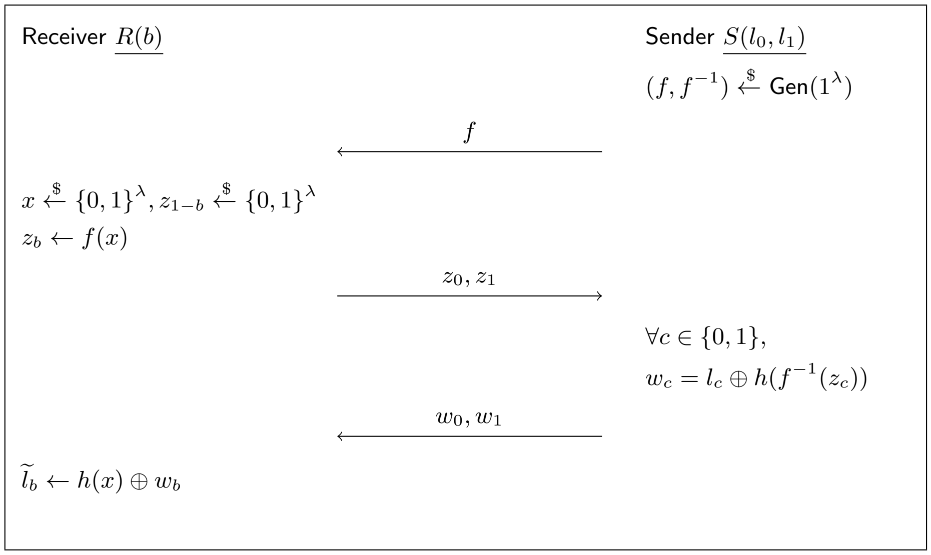
Receiver $\underline{R(b)}$
Sender $\underline{S(l_0, l_1)}$
$(f, f^{-1}) \overset{\$}{\leftarrow} \mathsf{Gen}(1^\lambda)$
f
$x \overset{\$}{\leftarrow} \{0,1\}^\lambda, z_{1-b} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
$z_b \leftarrow f(x)$
z_0, z_1
$\forall c \in \{0,1\},$
$w_c = l_c \oplus h(f^{-1}(z_c))$
w_0, w_1
$\widetilde{l}_b \leftarrow h(x) \oplus w_b$

Fig. 1: The EGL OT protocol ([13]). Security holds against semi-honest receivers and malicious senders

Here $h(\cdot)$ 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 one-way property of f and the security of the hard-core predicate, R cannot distinguish w_{1-b} from random as long as z_{1-b} is randomly chosen.

Prior works [30] ([35] respectively) devised non-black-box (black-box respectively) approaches to deal with both malicious senders and receivers. When dealing with malicious senders, they still require certifiable TDPs. Without ceritifiability, challenges arise, which are highlighted below.

Main challenge: necessity of certification. In the above described semi-honest protocol, a malicious sender is free to deviate from the protocol. If the malicious sender sends a function f that is not a permutation, by simply looking at values z_0 and z_1, it could decide which one of the values is randomly sampled from the domain of the function, and which one is computed by evaluating f on a random point. Specifically, $\{x \overset{\$}{\leftarrow} \{0,1\}^\lambda : x\}$ and $\{x \overset{\$}{\leftarrow} \{0,1\}^\lambda : f(x)\}$ are distinguishable to the sender for such an f, thereby leaking the receiver's input. To see why this is true, let us consider an extreme case in which a malicious sender picks a function f in which half of the points in the domain of f all have the same image y. Such a malicious sender, upon receiving the values z_0, z_1, checks if there exists $d \in \{0,1\}$ such that $y = z_d$. If this is the case, then the malicious sender outputs d, otherwise it outputs a random bit. It is easy to see that such a malicious sender guesses the input of the receiver with the probability negligibly close to $1/2 + 1/4$. The natural approach to dealing with a malicious adversary is to force an adversarial party to prove honest behavior using a zero-knowledge proof. In fact, in the NIZK constructions based on certifiable TDPs, removing certification is non-trivial since it has direct bearing on the soundness. A cheating prover that picks a function f that is not a permutation can break the soundness of the NIZK. In this context, [1] proposed the first approach to avoiding certifiability. Their solution proposes a special purpose NIZK to prove that f is a trapdoor permutation over the full domain. Thus the prover, when sending over f also sends a special purpose proof that f is indeed a trapdoor permutation over the full domain. As mentioned earlier, these results were further extended to the partial domain setting by [23, 6] for a more restricted class of TDPs. Unfortunately, all the above solutions are in the common random string (CRS) model, and therefore not applicable in our setting. Following the above line of work, the natural idea could be to devise a zero-knowledge proof in the plain model whereby the sender proves that the function f is indeed a trapdoor permutation. However, as we discuss below, this runs into fundamental barriers. The main challenge in requiring the sender to send a zero-knowledge proof to the receiver, is the limitation on the number of rounds. Even in the four-round setting, the receiver sends its last message in the third round, and thereby must know by the end of the second round if f sent by the sender is a permutation. This would thereby require the sender to complete its zero-knowledge proof by the send round, but providing such a zero-knowledge proof in two rounds is impossible [22]. Another naïve solution to extend the techniques of [1] in the plain model, would be to run a *challenge-response* protocol. In this, the party that wants to check if a function f is a permutation (the receiver R in this case), upon receiving the function f from the party that wants to certify that f is a permutation (the sender S in this case), samples random values from the domain, evaluates them, and sends them to S. S then inverts the received values and sends them back to R. R now can check if the received values correspond to the values he sampled from the domain of the function. It is easy to see that if the function is not a permutation, then (with some probability) one of the evaluated points R sends to S has a multiple pre-images, and S has no way to determine which pre-image R picked, resulting in R rejecting the function. The problem with this approach is that it requires at least three rounds of communication.

And this is clearly unacceptable if we want to construct an OT protocol that overall consists of three (or even four) rounds.

Dual Witness Encryption (DWE). As alluded to in our result section, we will rely on a Dual Witness Encryption scheme for the languages (L_0, L_1), where L_1 is an NP language. A Dual Witness Encryption is described by an encryption algorithm and a decryption algorithm. The encryption algorithm takes as input an instance (either in L_0 or in L_1) and a message, and outputs a ciphertext ct. The decryption algorithm takes as input a ciphertext, an instance $x \in L_1$ and a witness for x, and returns a string. A DWE enjoys the following two properties:

Completeness: If the cipthertext ct is computed using an instance $x \in L_1$, then the decryption algorithm, on input x and a witness for x, efficiently outputs the plaintext of ct.

Hiding: If the the cipthertext ct is computed using an instance $x \in L_0$, then ct hides the plaintext in an information-theoretic sense.

Our main idea in a nutshell. We now show how to use our techniques to transform the EGL protocol of Figure 1 into a protocol that protects the input of the receiver against malicious senders relying on TDPs only. An honest receiver wants to prevent a cheating sender from being able to view z_0 and z_1 if the sender has not picked f honestly. To facilitate this intuition, the receiver encrypts, using the dual witness encryption using f as the instance, its messages (z_0 and z_1), and sends over the corresponding ciphertext to the sender. On the one hand, if the sender has indeed picked a function by running the generation algorithm Gen, then it can decrypt and obtain z_0 and z_1, on the other hand if the selected function has a lot of collisions, then the ciphertext will hide z_0 and z_1. But this only gives us a weak form of security against malicious senders since the f picked might not have a lot of collisions. The security is then amplified using a weak notion of *OT combiners*. More precisely, we use a $(1, k)$-combiner that takes as input k OT instantiations and outputs a secure OT against malicious senders as long as there is at least one OT that is secure against malicious sender. We note that for simulation based security, we will have to do some further work and add an additional round. This construction is already sufficient to obtain a 3-round OT protocol that retains its security against malicious senders and semi-honest receivers[8] relying on uncertified TDPs.

Constructing a DWE scheme for TDPs. We start with the construction of the main tool used in our work: a DWE that encrypts with respect to a function f. For simplicity, we will limit our discussion to a bit encryption scheme, with a natural extension to encryption of bit strings. The rough idea to encrypt a bit m, is to sample an element x from the domain, compute $y \leftarrow f(x)$, and generate the ciphertext to be $(y, x_j \oplus m, j)$, where x_j is the j-th bit of x for a randomly

[8] We provide privacy for the input of the receiver in the sense that a malicious sender cannot distinguishes between when the receiver is using the input 0 and when he is using the input 1.

sampled j. On the one hand, if f was indeed a permutation, generated alongside the corresponding trapdoor f^{-1} (that can be obtained from the randomness used to ran the generation algorithm), one can decrypt the ciphertext. On the other hand, if f is not a permutation, then with some probability y has a collision, and thereby there exists at least another $x' \neq x$ such that $f(x') = f(x) = y$. Hence, with probability $1/n$ x and x' differ at the j-th position (where n is the size of x), thereby hiding m since the decryptor has no way to tell whether x or x' was used in the encryption. Of course, this only achieves a weak notion of security, that needs to be amplified. In order to amplify the security, we want to increase the likelihood of sampling an x such that $f(x)$ has a collision. We take the natural approach and additively secret share m as $m \leftarrow m_1 \oplus \cdots \oplus m_{\mathsf{q}}$, for an appropriate parameter q, and repeat the above strategy of encryption, with fresh randomness for each m_i separately. Now, when f is not a permutation, as long as at least one of the m_i remains hidden, m remains information theoretically hidden. In the technical section, we elaborate on this idea, and discuss the appropriate parameters required to guarantee security.

Towards a simulation-based secure construction. To obtain a complete solution (i.e., a protocol that is simulation based secure against malicious senders and receivers), we integrate the above idea in the [35] construction. However, doing so creates further challenges. The remainder of the section is dedicated to our solution, and how we tackle the challenges that arise.
Let us now look at our solution is more detail.

The ORS [35] methodology. The starting point for our protocol is the black-box OT protocol presented in [35]. Their protocol is constructed in two steps. In the first step, they construct a black-box OT protocol that is one-sided simulatable, i.e. the protocol is fully simultable against a malicious receiver, but only satisfies input indistinguishability against a malicious sender. In the second step, they then provide a general transformation that allows one to go from *one sided simulatable* OT to *fully simulatable* OT in a black-box manner. Since we can directly use their transformation in the second step, we limit our discussion to the construction of a *one sided simulatable* OT protocol. Looking back at our description of the semi-honest three-round oblivious transfer protocol, if we are to consider a fully malicious receiver $R^\star$ then this protocol is already no longer secure. Indeed $R^\star$ could just compute $z_{1-b} = f(y)$ picking a random $y \xleftarrow{\$} \{0,1\}^\lambda$. In this way $R^\star$ can retrieve both the inputs of the sender l_0 and l_1. In [30] the authors solve this problem by having the parties engage in a coin-flipping protocol such that the receiver is forced to set at least one of 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 [35], 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 [30] is one-sided simulatable. The main drawback in the above approach, addressed in [35], is that the use of a WI scheme requires using the commitment scheme in a non-black-box manner. Instead, in [35] the authors propose an approach that makes only black-box use of the underlying primitives. The main insight in [35] was to recast the problem in terms of equivocal and binding commitments, and having the output of the coin-flipping to be a pair of strings (z_0, z_1).

1. Receiver R, on secret input b, chooses random strings r_0 and r_1. R then sends across commitments $\mathtt{com}_0, \mathtt{com}_1$ such that $\mathtt{com}_{1-b}$ is a commitment to r_{1-b} while $\mathtt{com}_b$ is an equivocal commitment. R now proves that one of the commitments is binding.
2. The sender S then samples a trapdoor permutation from the family $f, f^{-1} \leftarrow \mathsf{Gen}(1^\lambda)$, and sends f to R. S also additionally samples a random string r, and sends it over to R.
3. R will now choose its decommitments to send to S. For $\mathtt{com}_{1-b}$ it will decommit to r_{1-b}, but for $\mathtt{com}_b$, R does the following. R sample $x \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and computes $z_b \leftarrow f(x)$, and sets $r_b \leftarrow z_b \oplus r$. R now decommits $\mathtt{com}_b$ to r_b, and both decommitments are sent to S.
4. S on receiving the decommitments, checks if they are valid before proceeding. It then sets $z_a \leftarrow r_a \oplus r$ for $a \in \{0,1\}$, and sends (w_0, w_1) to the receiver where $w_a \leftarrow \ell_a \oplus h(f^{-1}(z_a))$.

Since one of the commitments are guaranteed to be binding from the soundness of the proof, the receiver can only equivocate one of the strings, and thereby knows the pre-image to only one of the strings. From the above description, the main technical contributions of [35] are to realize the above protocols in a black-box manner using the commit-and-prove protocol due to [32]. The above description is sufficient to discuss the main ideas and challenges underlying our work, for a more detailed discussion of [35] we refer the reader to the technical sections. Given the description of the above protocol, and equipped with the dual witness encryption, the natural approach is for the receiver to encrypt its decommitments sent in the third round using the function f sent by the sender. This seems to work as a valid defense against a malicious sender, but an unwanted consequence of this modification is that simulation now fails for a malicious receiver. Let us see why this is the case.

Defending against a malicious receiver. Consider an execution of the simulator with a malicious receiver. At some point during the simulation, the simulator will receive the encrypted messages from the receiver, and must proceed with the simulation. But just from looking at the ciphertext, it does not know if the ciphertexts contain legitimate decommitments, or some arbitrary values. Why is this a problem? In the real execution of the OT protocol, an honest sender, having picked f to be a permutation, will decrypt the ciphertexts and abort if the ciphertexts do not decrypt to a legitimate decommitment. Therefore, in order to avoid a trivial distinguisher, the simulator must also perform this check. One natural way would be for the simulator to mimic the honest sender's behavior and decrypt the ciphertext, and then decide the appropriate action from the decrypted value. Unfortunately, this strategy does not work, and we illustrate why this would be a problem. In the above protocol, we said that the intuitive reason for the receiver not to learn l_{1-b} is that it does not know the pre-image of the random string, and thereby can do no better than guessing the hardcore predicate of the pre-image. To formalize this in the proof, we need to make a redution to security of the hardcore bit of f. Such a reduction receives only the function f and but must be able to complete the interaction against the malicious receiver. And importantly, must do so without knowledge of the randomness ρ used to generate f. A consequence of this is that decrypting is no longer an option since the reduction does not have ρ. In essence, if we have to decrypt to check, then we are breaking the security of the hardcore predicate. One way to get around decrypting, is to have some sort of "public check" such as a witness indistinguishable proof of knowledge as done in [30]. But the trivial application of this approach results in a non-black-box use of the underlying primitives, which we cannot afford to do. And indeed, it is unclear how one would prove honest behavior in such a scenario in a black-box manner. Taking a step back, we are seemingly deriving two distinct security properties from the function f: (i) for the security of the hardcore predicate against a malicious receiver; and (ii) hiding of the DWE scheme if f is not a permutation. The issue then is that when we want to rely on the security of the hardcore predicate, we do not care for the ciphertext to be hiding, since we are guaranteed that the function f used in the reduction, is a permutation. This seems to indicate that, while the current construction ties both these security properties, it does not necessarily have to be the case. Our approach is to decouple the above properties in a surprisingly simple manner. We use now two functions, an inner f_{OT} for the OT (and security of the hardcore predicate), and an outer f_{dWE} for the DWE scheme. The sender now samples the two functions along with the corresponding trapdoors. As before, the functions are sent to the reciever in the second round. The receiver then uses f_{OT} to compute z_b and uses f_{dWE} to encrypt the decommitment. This solves the issue indicated above since the reduction can decrypt without breaking the security of the hardcore predicate. This means that we can now reduce the security of the scheme to the security of the function f_{OT}. But now, a malicious sender could choose f_{dWE} to be a permutation, while choosing f_{OT} maliciously. We seem to have lost the advantage of using the DWE scheme. Our final solution is to stick to the idea of using two functions. But instead of fixing the roles of the two functions, allow the receiver to determine the roles of the corresponding function. As mentioned before, this provides only a weak guarantee and is amplified through the use of OT combiners. While we have described the main ideas underlying the construction, implementing these ideas involve further work, and we refer the reader to the relevant technical sections for the details.

2.1 Related Work

Oblivious Transfer. As stated earlier, oblivious transfer (OT) plays a fundamental role in cryptography and has a large body of work starting with [13]. We restrict ourselves to relevant works focusing on the round optimality of OT. In the random oracle model, [34] constructs a two-round OT protocol with indistinguishability based security against malicious receivers, but simulation security against malicious senders. In the CRS model, [37] constructs a fully maliciously secure two-round protocol. Moving to the relevant setting of the plain model, [30] showed that four rounds are necessary for a fully maliciously secure OT protocol with black-box simulation, and further proved that this was tight by constructing a four round OT protocol. The subsequent work [35] improved this construction by making only black-box use of the underlying primitives. In [26] the authors propose a weaker notion of trapdoor permutations whose permutation domains are polynomially dense (i.e., contain polynomial fractions of all strings of a particular length), and shows that these are sufficient to obtain semi-honest OT. Unfortunately, it does not seem that the construction of [26] would work against malicious senders, as the security of the protocol relies on the trapdoor function having a specific structure (i.e., being polynomially dense) that needs to be certifiable.

Round-complexity of 2PC. Studying the round complexity for secure computation has been the focus of many works in the past years. Whereas for unconditional security it is inherent to have protocol that are non-constant round [2, 8, 12], for the computational case it was showed that three rounds are sufficient to achieve security against semi-honest adversaries [40, 41], and subsequently [33, 31] showed constant round protocols for the case of malicious adversaries. In [30] the authors show that five rounds are necessary and sufficient to compute any two-party functionality where both parties can get the output (with black-box simulation)[9]. This result was later improved in [35] by showing how to obtain a 5-round protocol with black-box use of the underlying certifiable enhanced trapdoor permutations. In [16] the authors consider the case where the parties have a simultaneous message exchange channel[10] available and show that four rounds are necessary and sufficient to do secure computation assuming 3-robust non-malleable commitments. A followup work [10] showed how to obtain a four-round secure protocol when a simultaneous message exchange channel is available under the assumption of enhanced certifiable trapdoor permutations. We remark that in this paper we do not assume simultaneous message exchange channels.

Certifying trapdoor permutations. We have already mentioned some relevant works in this area and we now extend our discussion. [39, 23] discuss the security of the 1-out-of-k oblivious transfer protocol [13] which is based on trapdoor permutations, noting that its security is compromised in the case of partial-domain

[9] In this work we only refer to black-box simulation.

[10] In this model everyone can send messages at the same time.

trapdoor functions (when $k \geq 3$). [39, 23] then show how doubly enhanced trapdoor functions can be used to overcome this issue. Clearly, the problem of certifying trapdoor permutations does not arise when only semi-honest parties are considered (like in the case of semi-honest OT), but it is fundamental in the case of malicious adversary. This problem, for the case of secure computation, has been circumvented in [30, 35, 16, 10] by simply using certifiable trapdoor permutations. That is, by using trapdoor permutations equipped with a verification algorithm that can be used to check if a function is a permutation or not. The problem of getting rid of the certifiability property has been studied mostly for the case of NIZK in the shared random string model [1]. Recently [6] has studied additional certifiable properties that allows recognizing elements in the domain, as well as uniformly sample from it even for illegitimate functions, and show that some of these properties are necessary to apply the results of [1] to obtain a secure NIZK even for maliciously sampled trapdoor functions.

2.2 Organization of the Paper

In the next section we provide the fundamental background required to read our paper. We dedicate Section 4 to defining the notion of dual witness encryption, providing a few examples for the languages of DH tuples and QR tuples. In Section 5 we show how to instantiate a DWE for the language of non-TDPs. We devote Sections 6 and 7 to our 4-round OT protocol secure against malicious adversaries, and Section 8 to our round-optimal 2-PC protocol. For the formal construction and proofs of our 3-round OT protocol we refer the reader to the full version.

3 Background

Notation. 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, $x \overset{\$}{\leftarrow} Q$ denotes a sampling of x from Q with uniform distribution. We use "$=$" to check equality of two different elements (i.e. $a = b$ then...), "$\leftarrow$" as the assigning operator (e.g. to assign to a the value of b we write $a \leftarrow$ b). and $:=$ to define two elements as equal. 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* $\mathcal{R}$ (or *polynomial relation*, in short) is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x, w) in $\mathcal{R}$ can be decided in time polynomial in $|x|$. For $(x, w) \in \mathcal{R}$, we call x the *instance* and w a *witness* for x. For a polynomial-time relation $\mathcal{R}$, we define the NP-language $L_{\mathcal{R}}$ as $L_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$. Analogously, unless otherwise specified, for an NP-language L we denote by $\mathcal{R}_{\mathsf{L}}$ the corresponding polynomial-time relation (that is, $\mathcal{R}_{\mathsf{L}}$ is such that $L = L_{\mathcal{R}_{\mathsf{L}}}$).

When it is necessary to refer to the randomness r used by and algorithm A we use the following notation: $A(\cdot; r)$. We assume familiarity with the notion of computational and statistical indistinguishability, sigma-protocols and with the DDH assumption. We refer to the full version for the formal definitions.

3.1 Injective TDFs and TDPs

In this section we define the notion of trapdoor function following mostly the notation proposed in [6].

Definition 1 (Trapdoor function). *A family of one-way trapdoor functions, or TDFs, is a collection of finite functions, denoted $f_\alpha : \{D_\alpha \to R_\alpha\}$, accompanied by* PPT *algorithms* Gen, *S_D (domain sampler), S_R (range sampler) and two (deterministic) polynomial time algorithms* Eval *(forward evaluator) and* Inv *(backward evaluator) such that the following conditions hold.*

1. *On input 1^λ, the algorithm* Gen *selects a random index α of a function f_α, along with a corresponding trapdoor* $\mathtt{td}$.
2. *On input α, algorithm S_D samples an element from domain D_α.*
3. *On input α, algorithm S_R samples an image from the range R_α.*
4. *On input α and any $x \in D_\alpha$, $y \leftarrow \mathsf{Eval}(\alpha, x)$ with $y = f_\alpha(x)$.*
5. *On input $\mathtt{td}$ and any $y \in R_\alpha$, $\mathsf{Inv}(\mathtt{td}, y)$ outputs x such that $\mathsf{Eval}(\alpha, x) = y$.*

The standard hardness condition refers to the difficulty of inverting f_α on a random image, sampled by S_R or by evaluating Eval *on a random pre-image sampled by S_D, when given only the image and the index α but not the trapdoor* $\mathtt{td}$. *That is, let $I_0(1^\lambda)$ denote the first element in the output of $\mathsf{Gen}(1^\lambda)$ (i.e., the index); then, for every polynomial-time algorithm $\mathcal{A}$, it holds that:*

$$\Pr[(\alpha \overset{\$}{\leftarrow} I_0(1^\lambda); x \overset{\$}{\leftarrow} S_D(\alpha); y \leftarrow \mathsf{Eval}(\alpha, x), x' \overset{\$}{\leftarrow} \mathcal{A}(\alpha, y) : \mathsf{Eval}(\alpha, x') = y] \leq \nu(\lambda). \tag{1}$$

Or, when sampling an image directly using the range sampler:

$$\Pr[(\alpha \overset{\$}{\leftarrow} I_0(1^\lambda); y \overset{\$}{\leftarrow} S_R(\alpha); x' \overset{\$}{\leftarrow} \mathcal{A}(\alpha, y) : \mathsf{Eval}(\alpha, x') = y] \leq \nu(\lambda). \tag{2}$$

Additionally, it is required that, for any $\alpha \overset{\$}{\leftarrow} I_0(1^\lambda)$, the distribution sampled by S_R should be close the distribution sampled by $\mathsf{Eval}(S_D(\alpha))$. In this context we require the two distributions be computationally indistinguishable. We note that this requirement implies that the two hardness requirements given in equations 1 and 2 are equivalent. The issue of closeness of the sampling distributions is discussed further at the end of this section. If f_α is injective for all $\alpha \overset{\$}{\leftarrow} I_0(1^\lambda)$, we say that our collection describes an injective trapdoor function family, *or iTDFs (in which case $\mathsf{Inv}(\mathtt{td}, \cdot)$ inverts any images to its sole pre-image). If additionally D_α and R_α coincide than for any $\alpha \overset{\$}{\leftarrow} I_0(1^\lambda)$, the resulting primitive is a trapdoor permutation. If for any $\alpha \overset{\$}{\leftarrow} I(1^\lambda)$, $S_D = \{0,1\}^{\mathsf{poly}(\lambda)}$, that is, every* poly*-bit string describes a valid domain element, we say the function is full domain. Otherwise we say the domain is partial.*

Definition 2 (Hard-Core Predicate). *h is a* hard-core predicate *for f_α if its value is hard to predict for a random domain element x, given only α and $f_\alpha(x)$. That is, if for any* PPT *adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\Pr[(\alpha \overset{\$}{\leftarrow} I_0(1^\lambda); x \overset{\$}{\leftarrow} S_D(\alpha); y \overset{\$}{\leftarrow} \mathsf{Eval}(\alpha, x), h(x) \leftarrow \mathcal{A}(\alpha, y)] \leq 1/2 + \nu(\lambda).$$

Enhancements. Goldreich [19] suggested the notion of enhanced TDPs, which can be used for cases where sampling is required to be available in a way that does not expose the pre-image. We recall the notion of *enhanced injective TDF* proposed in [6] that extends the definition proposed by Goldreich to the case of injective TDF (where the domain and range are not necessarily equal).

Definition 3 (Enhanced injective TDF [19]). *let $\{f_\alpha : D_\alpha \to R_\alpha\}$ be a collection of injective TDFs, and let S_D be the domain sampler associated with it. We say that the collection is* enhanced *if there exists a range sampler S_R that returns a random sample out of R_α, and such that, for every polynomial-time algorithm $\mathcal{A}$, it holds that*

$$\text{Prob}\left[\ (\alpha \overset{\$}{\leftarrow} I_0(1^\lambda); y \overset{\$}{\leftarrow} S_R(\alpha; r); x' \overset{\$}{\leftarrow} \mathcal{A}(\alpha, r) : \mathsf{Eval}(\alpha, x') = y\ \right] \leq \nu(\lambda).$$

Definition 4 (Enhanced Hard-Core Predicate [21]). *let $\{f_\alpha : D_\alpha \to R_\alpha\}$ be an enhanced collection of injective TDFs with domain sampler S_D and range sampler S_R. We say that the predicate h is an enhanced hard-core predicate of f_α if it is computable in* PPT *time and for any* PPT *adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\Pr[(\alpha, \mathtt{td}) \overset{\$}{\leftarrow} \mathsf{Gen}(1^\lambda); r \overset{\$}{\leftarrow} \{0,1\}^\lambda; y \leftarrow S_R(\alpha; r); x \leftarrow \mathsf{Inv}(\mathtt{td}, y); \mathcal{A}(\alpha, r) = h(\alpha, x)] \leq 1/2 + \nu(\lambda)$$

or equivalently, if the following two distribution ensembles are computationally indistinguishable:

$$\{(\alpha, r, h(\alpha, \mathsf{Inv}(\mathtt{td}, S_R(\alpha, rw\alpha, \mathtt{td}) \overset{\$}{\leftarrow} \mathsf{Gen}(1^\lambda), r \overset{\$}{\leftarrow} \{0,1\}^\star\}$$
$$\{(\alpha, r, u) : \alpha \overset{\$}{\leftarrow} I_0(1^\lambda), r \overset{\$}{\leftarrow} \{0,1\}^\star, u \overset{\$}{\leftarrow} \{0,1\}\}$$

Additional Properties. We define multiple notions of certifiability for trapdoor functions, where each requires the existence of a general prover and verifier protocol for the function family. Let $f_\alpha : \{D_\alpha \to D_\alpha\}$ be a trapdoor permutation family, given by $(\mathsf{Gen}, S, \mathsf{Eval}, \mathsf{Inv})$ (where $S = S_R = S_D$), we now define the following properties.

Efficiently recognizable domain: that is, there exists a polynomial-time algorithm $\mathsf{R_D}$ which, for any index α and any string $x \in \{0,1\}^*$, accepts on (α, x) if and only if $x \in D_\alpha$. In other words, D_α is defined as the set of all strings x such that $\mathsf{R_D}(\alpha, x)$ accepts.

Efficiently sampleable domain: that is, there exists a PPT algorithm S_{DR} that on input α outputs a pair of (x, r) such that $\mathsf{Eval}(\alpha, x) = S(\alpha; r)$ where x is sampled uniformly in D_α.

Efficiently sampleable range: that is, for any index α and $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$, $S(\alpha; r)$ samples uniformly in D_α.

We stress that these properties should hold with respect to any α, including ones that were not generated by running $\mathsf{Gen}(1^\lambda)$. We also note that despite the similarities between the notions of doubly enhancement and efficiently sampleable domain, these two are incomparable. The notion of efficiently sampleable domain just requires the existence of a sampling algorithm that samples uniformly in D_α even for a maliciously chosen α, and it puts *no requirements* of one-wayness. Note that any trapdoor permutation family with full domain trivially enjoys all the properties listed above (one example is given by the candidate trapdoor permutation proposed in [38]). We show how to obtain a secure 2-party computation that relies on injective enhanced trapdoor permutations that have efficiently sampleable range and domain in a black-box way (note that we put no requirements on the certifiability of the injectivity). We finally recall that previous works required the existence of the same samplers even in the case of certifiable TDPs.

3.2 Commit-and-Open Protocols

In [15] the authors provide the definition of 3-round *commit-and-open* protocols. In this the prover (committer) has two inputs $m_0, m_1 \in \mathcal{M}$ and a bit $b \in \{0,1\}$ (we denote with $\mathcal{M}$ the message space of the commitment scheme). Informally, the message m_b is fixed in the first round of the protocol, and the message m_{1-b} can be decided in the last round where the messages (m_0, m_1) are revealed to the verifier (receiver). More formally, a commit-and-open protocol is a tuple of PPT algorithms $\Pi_{\mathsf{c\&o}} := (\mathsf{P} := (\mathsf{P}_0, \mathsf{P}_1), \mathsf{V} := (\mathsf{V}_0, \mathsf{V}_1))$ specified as follows. The algorithm P_0 takes as input m_b and outputs a string $\gamma \in \{0,1\}^\star$ and auxiliary state information $\alpha \in \{0,1\}^\star$. The algorithm V_0 outputs a random string $\beta \xleftarrow{\$} \mathcal{B}$ (where $\mathcal{B}$ represents the message space of the valid second rounds for $\Pi_{\mathsf{c\&o}}$). The algorithm P_1 takes as input $(\alpha, \beta, \gamma, m_{1-d})$ and outputs a string $\delta \in \{0,1\}^\star$. The deterministic algorithm V_1 takes a transcript $(\gamma, \beta, (\delta, m_0, m_1))$ and outputs a bit. Following [15], we denote with $< \mathsf{P}(m_0, m_1, b), \mathsf{V}(1^\lambda) >$ an execution of P where P uses (m_0, m_1, b) as input, and denote with $T := (\gamma, \beta, (\delta, m_0, m_1))$ the transcript obtained in this execution. We say that P satisfies completeness if honestly generated transcripts are always accepting (i.e., V_1 outputs 1).

Definition 5 (Secure commit-and-open protocol. [15]). *We say that a 3-round protocol $\Pi_{\mathsf{c\&o}}$ is secure if it enjoys completeness and satisfies the following properties.*

Existence of Committing Branch: *for every* PPT *malicious prover* $\mathsf{P}^\star := (\mathsf{P}_0^\star, \mathsf{P}_1^\star)$ *there exists a negligible function ν such that*

$$\Pr[\mathsf{V}_1(T) = 1 \text{ and } \mathsf{V}_1(T') = 1 \text{ and } m_0 \neq m_0' \text{ and } m_1 \neq m_1' : (\gamma, \alpha) \xleftarrow{\$} \mathsf{P}_0^\star, \\ \beta, \beta' \xleftarrow{\$} \mathsf{V}_0, (\delta, m_0, m_1) \xleftarrow{\$} \mathsf{P}_1^\star(\alpha, \beta), (\delta', m_0', m_1') \xleftarrow{\$} \mathsf{P}_1^\star(\alpha, \beta')] \leq \nu(\lambda)$$

where $T := (\gamma, \beta, (\delta, m_0, m_1))$ and $T' := (\gamma, \beta', (\delta', m_0', m_1'))$, and where the probability is taken over the random coin tosses of P *and* V.

Committing Branch Indistinguishability: *for all* PPT *malicious verifier* $\mathsf{V}^\star$, *and for all messages $m_0, m_1 \in \mathcal{M}$, we have that the following are indistinguishable*

$$\{T \;:\; T \overset{\$}{\leftarrow} < \; P(m_0, m_1, 0), \mathsf{V}^\star(1^\lambda) \; >\}_{\lambda\in\mathbb{N}}$$
$$\{T \;:\; T \overset{\$}{\leftarrow} < \; P(m_0, m_1, 1), \mathsf{V}^\star(1^\lambda) \; >\}_{\lambda\in\mathbb{N}}$$

The authors of [15] show that one of the protocols proposed in [35] that relies on statistically binding and computationally hiding commitment (and it is black-box in the use of the underlying primitives) satisfies the above definition. Since statistically binding and computationally hiding commitments can be constructed using one-to-one one way-functions in a black-box manner then there exists a secure commit-and-open protocol that uses the underlying one-way function is a black-box way. We refer to [15] for more discussion on the notion of commit-and-open and for its black-box instantiation from one-to-one one-way-functions.

3.3 Oblivious Transfer and 2-PC

Here we follow [35]. 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 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 $F_{\mathcal{OT}}$ and oracle access to $S^\star$. We also consider the weaker definition of OT introduced in [35] which is referred as *one-sided simulatable OT*. In this 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. That is, we require that for any PPT malicious sender $S^\star$ the view of $S^\star$ executing Π with the receiver R playing with bit 0 is computationally indistinguishable from the view of $S^\star$ where R is playing with the 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.

Definition 6 ([35]). *Let $F_{\mathcal{OT}}$ be the Oblivious Transfer functionality as described previously. 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 7 ([35]). *A protocol* Π *securely realizes* $F_{\mathcal{OT}}$ *with fully simulatability if* Π *is one-sided simulatable and additionally for every non-uniform* PPT *adversary* $S^\star$ *controlling the sender in the real model, there exists a non-uniform* PPT *adversary* Sim *for the ideal world such that*

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

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

In this work we also consider the notion of parallel OT, which is the same as the previous definition, except that the sender has multiple pairs of inputs and the receiver has multiple bits.

Secure Two-Party Computation [35] Let $F(x_1,x_2)$ be a two-party functionality run between parties P_1 holding input x_1 and P_2 holding input x_2. In the ideal world, P_i with ($i \in \{1,2\}$) sends its input x_i to the F and obtains only $y = F(x_1,x_2)$. We say that a protocol Π securely realizes F if the view of any malicious $P_i^\star$ executing Π with an honest P_j with $i \neq j$ combined with the output of P_j (if any) can be simulated by a PPT simulator that has only access to F and has oracle access to $P_i^\star$.

4 Dual Witness Encryption (DWE)

A Dual Witness Encryption scheme for the languages L_0, L_1 with $L_0, L_1 \subseteq \{0,1\}^\star$ is equipped with two PPT algorithms: Enc and Dec. Enc takes as input $x \in \{0,1\}^\lambda$, a message $m \in \{0,1\}^\lambda$ and outputs $ct \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Dec takes as input $x \in \{0,1\}^\lambda$,$w \in \{0,1\}^\lambda$, $ct \in \{0,1\}^{\mathsf{poly}(\lambda)}$ and outputs a message $m \in \{0,1\}^\lambda \cup \{\perp\}$.

Definition 8. *A Dual Witness Encryption scheme* $\mathsf{PK\text{-}IBS} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for the languages* (L_0, L_1) *is secure if it enjoys the following properties.*

Completeness: $\Pr[m \leftarrow \mathsf{Dec}(x, w, \mathsf{Enc}(x, m)) = 1 : (x, w) \in \mathcal{R}_{L_1}] \geq 1 - \nu(\lambda)$.
Hiding: *For any adversary $\mathcal{A}$ and for any $x \in L_0$ the following holds:*

$$\Pr[b \overset{\$}{\leftarrow} \{0,1\}; (m_0, m_1) \leftarrow \mathcal{A}(x) \wedge b \leftarrow \mathcal{A}(\mathsf{aux}, \mathsf{Enc}(x, m_b))] < \nu(\lambda)$$

4.1 DWE for the languages of DH and QR Tuples.

In this section we show how to construct a DWE for the languages of DH and and QR tuples. We do not need these constructions to build our OT and 2PC protocols, we only want to show that our primitive can be instantiated also with respect to other languages. The two constructions rely on similar ideas, hence, we provide the details only for the construction for DH tuples. Our constructions are based on the sigma-protocol for the language of the DH and QR tuples and on some observations made in [11, 5] on these sigma protocols. Following [11], we recall the well-known Sigma protocol $\Sigma_{DH} = (\mathcal{P}, \mathcal{V})$ for the language $L_0 := \{(g, h, U, V) : \exists \alpha \text{ s.t. } U = g^\alpha \text{ and } V = h^\alpha\}$. On common input $T = (g, h, U, V)$, and honest prover's private input α such that $U = g^\alpha$ and $V = h^\alpha$, the following steps are executed. We denote the size of the group $\mathcal{G}$ by q.

- $\mathcal{P}$ picks $r \in \mathbb{Z}_q$ at random and computes and sends $A := g^r$, $B := h^r$ to $\mathcal{V}$;
- $\mathcal{V}$ chooses a random challenge $c \in \{0,1\}$ and sends it to $\mathcal{P}$;
- $\mathcal{P}$ computes and sends $z = r + \alpha \cdot c$ to $\mathcal{V}$;
- $\mathcal{V}$ accepts if and only if $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$.

In [11] the authors observe that the above protocol has the following interesting property. There exists a PPT algorithm $\mathsf{ChallExt}$ that on input a first round $a = (A, B)$ of Σ_{DH}, a non-DH tuple T and γ such that $h = g^\gamma$, outputs the only valid second round $c \in \{0,1\}$ (if any exists) such that there is some z that would make the verifier to (mistakenly) accept the transcript (a, c, z) with respect to the instance T. The algorithm $\mathsf{ChallExt}$ works as follows. Let $T = (g, h, X, W)$ be a non-DH tuple such that $X = g^\alpha$, $W = h^\beta$, $\alpha \neq \beta$ and $h = g^\gamma$. Upon input $(T = (g, h, X, W), a, \gamma)$, algorithm $\mathsf{ChallExt}$ parses a as (A, B), and if $A^\gamma = B$ then it outputs 0, else it outputs 1. Note that when the first round of Σ_{DH} corresponds to a DH tuple, (*i.e.*, $A^\gamma = B$) and T is not a DH tuple, then the only c that would make true the conditions $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$ is $c = 0$. Instead, if (g, h, A, B) does not represent a DH tuple (*i.e.*, $A^\gamma \neq B$) then there exists z such that $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$ if and only if $c = 1$. In what follows, we make use of this special property of Σ_{DH}, and we refer to $\mathsf{ChallExt}$ as the *bad-challenge extractor*. The same holds true for the classical Sigma protocol for QR [25] (along the lines of the full version of [5, Sec. 6.2]). The above observation, together with the fact that Σ_{DH} is SHVZK immediately yields to a DWE for the languages (L_0, L_1) where $L_1 = \{0,1\}^\star - L_0$, and where the NP-relation associated to L_1 is $\mathcal{R}_{L_1} := \{(g, h, X, W), \gamma : \; h = g^\gamma \text{ and } W \neq X^\gamma\}$.

In more detail, the encryption algorithm works by running the SHVZK simulator for Σ_{DH} on input $T \in L_0 \cup L_1$ and the message to be encrypted $m \in$

$\{0,1\}$. The output of the SHVZK algorithm corresponds to $(A := g^{z-\alpha m}, B := h^{z-\beta m}, z)$. The output of our encryption algorithm then corresponds to (A, B).

If $T \in L_1$ (i.e., it is a non-DH tuple), then we can run the bad-challenge extractor $\mathsf{ChallExt}$ to reconstruct m in polynomial-time (note that the tuple (g, h, A, B) is DH only if $m = 0$). In the case when T is a DH tuple, then, by the completeness and the SHVZK properties of Σ_{DH}, (A, B) encodes no information on the message m. Indeed, it is alway possible to find a valid z that makes the transcript $(A, B), m, z$ accepting for any $m \in \{0,1\}$. For sake of completeness we now provide the formal description of our protocol, that we denote with $(\mathsf{Enc}^{\mathsf{NDH}}, \mathsf{Dec}^{\mathsf{NDH}})$.

- Let $m \in \{0,1\}$ be the message to be encrypted. The encryption algorithm $\mathsf{Enc}^{\mathsf{NDH}}$ takes as input the tuple $T = (g, h, X, W)$ and the message $m \in \{0,1\}$ and does the following steps.
  1. Sample $z \in \mathbb{Z}_q$ and compute $A \leftarrow \frac{g^z}{X^m}, B \leftarrow \frac{h^z}{W^m}$
  2. Output A, B.
- The algorithm $\mathsf{Dec}^{\mathsf{NDH}}$ takes as input $T \in L_1$, the ciphertext (A, B) and the witness γ such that $(T, \gamma) \in \mathcal{R}_{L_1}$, and outputs $\mathsf{ChallExt}(T, A, B, \gamma)$.

Theorem 3. $(\mathsf{Enc}^{\mathsf{NDH}}, \mathsf{Dec}^{\mathsf{NDH}})$ *is a secure black-box DWE scheme with message space* $\{0,1\}$ *for the languages* (L_0, L_1) *defined above, where the relation associated to* L_1 *is* $\mathcal{R}_{L_1}$.

DWE for all NP languages. If we do not care about the decryption algorithm being efficient (PPT), then the above approach can be extended to any NP language L that admits a sigma-protocol Σ. Indeed, if the instance used during the encryption is $x \notin L$, then the special soundness of Σ guarantees that for any first round of Σ there exists at most one challenge that would make the verifier to accept. This means that the first output of the SHVZK simulator of Σ on input x and the message $m \in \{0,1\}$ encodes m. Hence, an unbounded decryptor can easily compute it. On the other hand, when $x \in L$, then the first round of Σ (hence, the first output of the SHVZK simulator) information theoretically hides the message m (due to the completeness and the SHVZK properties of Σ).

5 Black-Box DWE for Trapdoor Permutations

A function $f_\alpha : D_\alpha \rightarrow D_\alpha$ is an ϵ-permutation if at most an ϵ fraction of the points in D_α have more than one pre-image (under f_α). More formally, we have the following.

Definition 9. *Let* $f_\alpha : \{D_\alpha \rightarrow D_\alpha\}$. *The* collision set *of* f_α, *denoted with* $C(f_\alpha)$, *is* $\{x_1 \in D_\alpha : \exists x_2 \in D_\alpha$ *s.t.* $x_1 \neq x_2$ *and* $\mathsf{Eval}(\alpha, x_1) = \mathsf{Eval}(\alpha, x_2)\}$. *Let* $\epsilon \in [0,1]$, *we call* f_α *an* ϵ-permutation *if* $|C(f_\alpha)| \leq \epsilon |D_\alpha|$.

We say that f_α is an *almost* permutation if it is an $\epsilon(n)$-permutation where ϵ is a negligible function and $n = |D_\alpha|$. Let $f_\alpha : \{D_\alpha \to D_\alpha\}$ be a collection of trapdoor permutations with efficiently sampleable range and domain accompanied by the algorithms $(\mathsf{Gen}, S, \mathsf{Eval}, \mathsf{Inv})$. We then define L as the language of trapdoor functions with efficiently sampleable range and domain that that have a collision set greater (or equal) than half of the entire domain. More formally, $L_0 = \{\alpha : |C(f_\alpha)| \geq 2^{-1}|D_\alpha|\}$. We also define L_1 as the set trapdoor function in the range of the generation algorithm Gen (i.e., $L_1 = \{\alpha : (\alpha, \mathtt{td}) \leftarrow \mathsf{Gen}(1^\lambda; r), r \in \{0,1\}^\lambda\}$) We provide a DWE scheme for the languages (L_0, L_1). Informally, this encryption scheme maintains the hiding of the encrypted message if the collision set of f_α is sufficiently large (i.e., f_α is *a lot* non-injective). Instead, if the function is generated using $\mathsf{Gen}(1^\lambda)$, then any message can be decrypted using the corresponding trapdoor (which is also an output of Gen and thus can be obtained from the randomness r, which represents the witness).

5.1 Our Constructions

We start by constructing a dual witness encryption scheme $(\mathsf{Enc}_1^{\mathsf{f}}, \mathsf{Dec}_1^{\mathsf{f}})$ for one-bit messages for the language (L_0, L_1) described above. Let f_α be a trapdoor permutation with efficiently sampleable range accompanied by the algorithms $(\mathsf{Gen}, S, \mathsf{Eval}, \mathsf{Inv})$ with domain (and range) of size 2^λ.

- Let $m \in \{0,1\}$ be the message to be encrypted, $\alpha \in L_1$, and $n := 2\lambda^2$. The encryption algorithm $\mathsf{Enc}_1^{\mathsf{f}}$ takes as input (α, m) and does the following steps.
  1. Compute a random secret sharing of m such that $m = m_1 \oplus \cdots \oplus m_n$.
  2. For $i \leftarrow 1, \ldots, n$ pick $x_i \xleftarrow{\$} S(\alpha)$ and compute $y_i \leftarrow f_\alpha(x_i)$.[11]
  3. For $i \leftarrow 1, \ldots, n$ parse x_i as $x_i^1 || \ldots || x_i^\lambda$, pick $j_i \xleftarrow{\$} \{1, \ldots, \lambda\}$ and compute $c_i \leftarrow m_i \oplus x_i^{j_i}$.
  4. Output $ct \leftarrow (j_i, y_i, c_i)_{i \in [n]}$.
- The algorithm $\mathsf{Dec}_1^{\mathsf{f}}$ takes as input α, r and a ciphertext ct_i, and executes the following steps.
  1. Compute $(\alpha, \mathtt{td}) \leftarrow \mathsf{Gen}(1^\lambda; r)$.
  2. Parse ct as $\big(j_i, y_i, c_i\big)_{i \in [n]}$.
  3. For $i = 1, \ldots, n$ compute $x_i \leftarrow \mathsf{Inv}(\alpha, \mathtt{td}, y_i)$, parse x_i as $x_i^1 || \ldots || x_i^n$ and compute $m_i \leftarrow c_i \oplus x_i^{j_i}$.
  4. Compute and output $m \leftarrow m_1 \oplus \cdots \oplus m_n$.

Theorem 4. $(\mathsf{Enc}_1^{\mathsf{f}}, \mathsf{Dec}_1^{\mathsf{f}})$ *is a secure black-box DWE scheme for the languages* (L_0, L_1) *with message space* $\{0,1\}$.

We refer to the full version for the formal proof of the theorem. We note that to obtain a DWE secure scheme $(\mathsf{Enc}^{\mathsf{f}}, \mathsf{Dec}^{\mathsf{f}})$ for messages of length $\kappa \in \mathbb{N}$ we can just run κ parallel executions of $(\mathsf{Enc}_1^{\mathsf{f}}, \mathsf{Dec}_1^{\mathsf{f}})$.

[11] To not overburden the notation we use f_α instead of $\mathsf{Eval}(\alpha, \cdot)$ as the evaluation algorithm hereafter in the paper.

DWE for *or* Statements For our OT constructions we use as a main tool a DWE for the languages $(L_0^{\mathsf{2f}}, L_1^{\mathsf{2f}})$ where $L_0^{\mathsf{2f}} := \{\alpha_0, \alpha_1 : |C(f_{\alpha_0})| \geq 2^{-1}|D_{\alpha_0}|$ or $|C(f_{\alpha_1})| \geq 2^{-1}|D_{\alpha_1}|\}$ and $L_1^{\mathsf{2f}} = \{\alpha_0, \alpha_1 : (\alpha_0, \mathtt{td}_0) \leftarrow \mathsf{Gen}(1^\lambda; r_0)$ and $(\alpha_1, \mathtt{td}_1) \leftarrow \mathsf{Gen}(1^\lambda; r_1), r_0, r_1 \in \{0,1\}^\lambda\}$. (we recall that we denote with $C(f_\alpha)$ the collision set of the function indexed by α). Informally, we require the semantic security of the encryption scheme to hold if *at least one* of the functions used as a part of the public-key has a collision set of sub-exponential size. Our scheme $(\mathsf{Enc}^{\mathsf{2f}}, \mathsf{Dec}^{\mathsf{2f}})$ works as follows.

- The encryption algorithm $\mathsf{Enc}^{\mathsf{2f}}$ on input $x := (\alpha_0, \alpha_1)$ and the message to be encrypted $m \in \{0,1\}^\kappa$ does the following steps.
  1. Run $\mathsf{Enc}^{\mathsf{f}}$ on input α_0 and m thus obtaining ct_0.
  2. Run $\mathsf{Enc}^{\mathsf{f}}$ on input α_1 and ct thus obtaining ct_1 and output ct_1
- The decryption algorithm $\mathsf{Dec}^{\mathsf{2f}}$ on input $x := (\alpha_0, \alpha_1)$, the witness $w := (r_0, r_1)$ and the ciphertext ct_1, executes the following steps.
  1. Compute $(\alpha_0, \mathtt{td}_0) \leftarrow \mathsf{Gen}(1^\lambda; r_0)$ and $(\alpha_1, \mathtt{td}_1) \leftarrow \mathsf{Gen}(1^\lambda; r_1)$.
  2. Run $\mathsf{Dec}^{\mathsf{f}}$ on input α_1, r_1, ct_1 and $\mathtt{td}_1$ thus obtaining ct_0.
  3. Run $\mathsf{Dec}^{\mathsf{f}}$ on input α_0, r_0 ct_0 and $\mathtt{td}_0$ thus obtaining m and output m.

Theorem 5. $(\mathsf{Enc}^{\mathsf{2f}}, \mathsf{Dec}^{\mathsf{2f}})$ *is a black-box DWE scheme for the languages* $(L_0^{\mathsf{2f}}, L_1^{\mathsf{2f}})$ *with message space* $\{0,1\}^\kappa$.

The proof in this case follow via standard hybrid arguments.

6 *Almost Secure* OT Protocol

In this section we show how to obtain a protocol $\Pi_{\mathcal{OT}} = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$ that securely realizes $F_{\mathcal{OT}}$ with one-sided simulation against any *weak* adversarial sender $S^\star_{\mathcal{OT}}$. Informally, we show that if the malicious sender $S^\star_{\mathcal{OT}}$ samples the trapdoor permutations used in the protocol in some particular ways then $\Pi_{\mathcal{OT}}$ is secure, otherwise we give no security guarantees. At a very high level our protocol works like the four-round one-side simulatable OT protocol proposed in [35]. As highlighted in the Introduction, in the ORS protocol the sender sends a trapdoor permutation f in the second round which is used by the receiver to compute the third round. In case that f is non-injective then a malicious sender, by just inspecting the third round sent by the receiver, could extract the receiver's input. In our protocol we try to avoid this attack by modifying the ORS protocol in two aspects: 1) the sender sends two trapdoor functions[12] in the first round and 2) the receiver samples a random bit to decide which function to use to run ORS and which function to use to run DWE scheme Π. Π is a DWE scheme that guarantees security if the trapdoor function used for the encryption has a lot of collisions, and it is used by the receiver to encrypt the third round of ORS. Unfortunately we cannot prove that this OT protocol is (in general) secure, but we can prove that it is secure if one of the following cases occurs.

[12] We need to send two pairs of functions, but for now we omit this since it is a technical detail that will be helpful in the security proof.

1. The malicious sender uses functions that are almost permutation. This comes with no surprise since in this case an execution of $\Pi_{\mathcal{OT}}$ looks like an execution of the ORS protocol.
2. The malicious sender uses functions that have a lot of collisions (exponentially many). In this case the security of the DWE scheme kicks in protecting all the information that are related to the ORS protocol that depends on the TDPs (i.e., the information that could leak the receiver's bit when the functions sampled by the sender are non-injective).

Despite this limitation, in Section 7 we show that the security enjoyed by $\Pi_{\mathcal{OT}}$ is (surprisingly) enough to obtain a secure OT protocol. We now provide a more detailed description of $\Pi_{\mathcal{OT}}$ and prove formally its *w*eak security in the case of malicious sender. Moreover, we show that $\Pi_{\mathcal{OT}}$ is secure against any PPT adversarial receiver under the standard simulation base security notion.

To construct $\Pi_{\mathcal{OT}}$ we make use the following tools.

1. A commit-and-open protocol $\Pi_{\mathsf{c\&o}} := (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V}_0, \mathsf{V}_1)$.
2. An enhanced trapdoor permutation with efficiently sampleable range and domain $\mathcal{F} := (\mathsf{Gen}, S, S_{DR}, f, f^{-1})$[13] with hard-core predicate h and domain (and range) of size 2^λ.
3. The DWE scheme $(\mathsf{Enc}^{\mathsf{2f}}, \mathsf{Dec}^{\mathsf{2f}})$ for the languages $(L_0^{\mathsf{2f}}, L_1^{\mathsf{2f}})$ described in Sec. 5.

We now give an informal description of our protocol and refer to Fig. 2 for the formal description.

Let $b \in \{0,1\}$ be the input of $R_{\mathcal{OT}}$ and $l_0, l_1 \in \{0,1\}$ be the input of $S_{\mathcal{OT}}$.

In the **first round** $R_{\mathcal{OT}}$ runs P_0 on input a string $r_{1-b} \xleftarrow{\$} \{0,1\}^\lambda$ thus obtaining the first round of the commit-and-open protocol $\Pi_{\mathsf{c\&o}}$.

In the **second round** $S_{\mathcal{OT}}$ picks a pair of random strings and samples four trapdoor permutations. That is, $S_{\mathcal{OT}}$ picks $R_0 \xleftarrow{\$} \{0,1\}^\lambda$, $R_1 \xleftarrow{\$} \{0,1\}^\lambda$, and for all $i, j \in \{0,1\}$ samples $\rho_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$, computes $(f_{i,j}, f_{i,j}^{-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \rho_{i,j})$. Then $S_{\mathcal{OT}}$ runs V_0 thus obtaining γ and sends $\{f_{i,j}\}_{i,j \in \{0,1\}}, \beta, R_0, R_1$ to $R_{\mathcal{OT}}$.

In the **third round** $R_{\mathcal{OT}}$ chooses a bit d and computes $(z', r') \xleftarrow{\$} S_{DR}(f_{\mathsf{d},b})$ and $r_b \leftarrow r' \oplus R_b$. Then $R_{\mathcal{OT}}$ computes the third round δ of $\Pi_{\mathsf{c\&o}}$ to open the commitment to the messages r_{1-b} (that is fixed in the first round) and r_b by running P_1 on input α, β, γ and r_b. In the end, $R_{\mathcal{OT}}$ encrypts the opening of $\Pi_{\mathsf{c\&o}}$ using the DWE scheme on input $(f_{1-\mathsf{d},0}, f_{1-\mathsf{d},1})$ and the message $\delta||r_0||r_1$ thus obtaining c and sends (c, d) to $S_{\mathcal{OT}}$.

In the **fourth round** $S_{\mathcal{OT}}$ decrypts c using the witness $\rho_{1-\mathsf{d},0}$ and $\rho_{1-\mathsf{d},1}$, thus obtaining the opening information of $\Pi_{\mathsf{c\&o}}$ represented by δ, r_0 and r_1. Then $S_{\mathcal{OT}}$ checks if (δ, r_0, r_1) represents a valid opening for $\Pi_{\mathsf{c\&o}}$ by running

[13] For convenience, we drop $(\mathsf{Eval}(\alpha,\cdot), \mathsf{Inv}(\alpha,\cdot))$ from the notation, and write $f(\cdot)$, $f^{-1}(\cdot)$ to denote algorithms $\mathsf{Eval}(f_\alpha,\cdot)$, $\mathsf{Inv}(f_\alpha, \mathtt{td}, \cdot)$ respectively, when f_α and $\mathtt{td}$ are clear from the context. We also use the function f_α instead of the index α as input of the algorithm S and S_{DR}.

V_1. If it is not, then $S_{\mathcal{OT}}$ stops and outputs $\perp$, otherwise she computes $\omega_0 \leftarrow f^{-1}_{\mathsf{d},0}(S(f_{\mathsf{d},0}, r_0 \oplus R_0))$ and $\omega_1 \leftarrow f^{-1}_{\mathsf{d},1}(S(f_{\mathsf{d},1}, r_1 \oplus R_1))$. Then for $j = 0, 1$, $S_{\mathcal{OT}}$ encrypts the input l_j via one-time pad using as a key the output of the hard-core predicate of $f_{\mathsf{d},j}$ on input ω_j thus obtaining W_j. $S_{\mathcal{OT}}$ then sends (W_0, W_1) to $R_{\mathcal{OT}}$ and stops.

In the **output phase**, $R_{\mathcal{OT}}$ computes and outputs $l_b = W^1_b \oplus h(f_{\mathsf{d},b}, z'_1)$.

In Fig. 2 we propose a formal description of the protocol.

Theorem 6. *If $\mathcal{F}$ is family of enhanced trapdoor permutations then 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* $\{\mathsf{REAL}_{\Pi_{\mathcal{OT}}, R^\star_{\mathcal{OT}}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda} \approx \{\mathsf{IDEAL}_{F_{\mathcal{OT}}, \mathsf{Sim}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda}$.[14]

We refer to the full version for the formal proof of the theorem.

Theorem 7. *For every non-uniform* PPT *adversary $S^\star_{\mathcal{OT}}$ controlling the sender, if one of the following holds with overwhelming probability*

1. *$f_{0,0}$ and $f_{0,1}$ and $f_{1,0}$ and $f_{1,1}$ are almost permutations or*
2. *$(f_{0,0}, f_{0,1}) \in L^{\mathsf{2f}}_0$ and $(f_{1,0}, f_{1,1}) \in L^{\mathsf{2f}}_0$.*

then $\{\mathsf{View}^{R_{\mathcal{OT}}}_{\Pi_{\mathcal{OT}}, S^\star_{\mathcal{OT}}(z)}(l_0, l_1, 0)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{View}^{R_{\mathcal{OT}}}_{\Pi_{\mathcal{OT}}, S^\star_{\mathcal{OT}}(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^\star}$

We refer the reader to the full version for the formal proof of the theorem. In the full version we also prove the following lemma that will be helpful hereafter. Before stating the lemma, we introduce some additional notations. We say that a value $y \in Y$ is *good* if there exists and is unique a value x such that $f_\alpha(x) = y$. We now denote with ${\mathsf{E_g}}^i$ the event in which a randomly sampled element from the range of f_i is *good* and prove this additional lemma.

Lemma 1. *For every non-uniform* PPT *adversary $S^\star_{\mathcal{OT}}$ controlling the sender, if one of the following holds with overwhelming probability*

1. Prob $\left[\ {\mathsf{E_g}}^{i,j}\ \right] \geq 1 - \nu(\lambda)\ \forall i, j \in \{0, 1\}$ *or*
2. Prob $\left[\ {\mathsf{E_g}}^{0,0}\ \right] < 2^{-1}$ *or* Prob $\left[\ {\mathsf{E_g}}^{0,1}\ \right] < 2^{-1}$ *and* Prob $\left[\ {\mathsf{E_g}}^{1,0}\ \right] < 2^{-1}$ *or* Prob $\left[\ {\mathsf{E_g}}^{1,1}\ \right] < 2^{-1}$

then $\{\mathsf{View}^{R_{\mathcal{OT}}}_{\Pi_{\mathcal{OT}}, S^\star_{\mathcal{OT}}(z)}(l_0, l_1, 0)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{View}^{R_{\mathcal{OT}}}_{\Pi_{\mathcal{OT}}, S^\star_{\mathcal{OT}}(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^\star}$

7 Secure OT from almost secure OT

In Theorem 7 we have showed that $\Pi_{\mathcal{OT}} = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$ guarantees that the input of the receiver is protected only in the case that at least one of the following properties holds:

1. $f_{0,0}$ and $f_{0,1}$ and $f_{1,0}$ and $f_{1,1}$ are almost permutations or

[14] We refer to Sec. 3.3 for a formal definition of $\mathsf{REAL}_{\Pi_{\mathcal{OT}}, R^\star_{\mathcal{OT}}(z)}$ and $\mathsf{IDEAL}_{F_{\mathcal{OT}}, \mathsf{Sim}(z)}$

$R_{\mathcal{OT}}(b)$ $\qquad\qquad\qquad$ $S_{\mathcal{OT}}(l_0, l_1)$

$r_{1-b} \xleftarrow{\$} \{0,1\}^\lambda$
$(\gamma, \alpha) \xleftarrow{\$} \mathsf{P}_0(r_{1-b})$

$$\xrightarrow{\quad\gamma\quad}$$

$\forall i, j \in \{0,1\}, \rho_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$
$(f_{0,0}, f_{0,0}^{-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \rho_{0,0})$
$(f_{0,1}, f_{0,1}^{-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \rho_{0,1})$
$(f_{1,0}, f_{1,0}^{-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \rho_{1,0})$
$(f_{1,1}, f_{1,1}^{-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \rho_{1,1})$
$\beta \xleftarrow{\$} \mathsf{V}_0(1^\lambda)$
$R_0 \xleftarrow{\$} \{0,1\}^\lambda, R_1 \xleftarrow{\$} \{0,1\}^\lambda$

$$\xleftarrow{\{f_{i,j}\}_{i,j\in\{0,1\}}, \beta, R_0, R_1}$$

$\mathsf{d} \xleftarrow{\$} \{0,1\}$
$(z', r') \xleftarrow{\$} S_{DR}(f_{\mathsf{d},b}),$
$r_b \leftarrow r' \oplus R_b$
$\delta \xleftarrow{\$} \mathsf{P}(\alpha, \beta, \gamma, r_b)$
$x := (f_{1-\mathsf{d},0}, f_{1-\mathsf{d},1})$
$c \xleftarrow{\$} \mathsf{Enc}^{\mathsf{2f}}(x, \delta || r_0 || r_1)$

$$\xrightarrow{\quad c, \mathsf{d}\quad}$$

$x := (f_{1-\mathsf{d},0}, f_{1-\mathsf{d},1})$
$w := (\rho_{1-\mathsf{d},0}, \rho_{1-\mathsf{d},1})$
$\delta || r_0 || r_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(x, w, c)$
if $\mathsf{V}_2(\gamma, \beta, \delta, r_0 || r_1) = 0$ **then**
  stop and output $\perp$
else continue as follows
$\omega_0 \leftarrow f_{\mathsf{d},0}^{-1}(S(f_{\mathsf{d},0}, r_0 \oplus R_0))$
$\omega_1 \leftarrow f_{\mathsf{d},1}^{-1}(S(f_{\mathsf{d},1}, r_1 \oplus R_1))$
$W_0 = l_0 \oplus h(f_{\mathsf{d},0}, \omega_0)$
$W_1 = l_1 \oplus h(f_{\mathsf{d},1}, \omega_1)$

$$\xleftarrow{\quad W_0, W_1\quad}$$

Output $l_b = W_b \oplus h(f_{\mathsf{d},b}, z')$

Fig. 2: Description of $\Pi_{\mathcal{OT}}$.

2. $(f_{0,0}, f_{0,1}) \in L_0^{\mathsf{2f}}$ and $(f_{1,0}, f_{1,1}) \in L_0^{\mathsf{2f}}$.

Moreover, Theorem 6 guarantees $\Pi_{\mathcal{OT}}$ is secure against malicious receivers. In this section we show that the above property is sufficient to obtain a one-sided simulatable OT by means of a compiler that takes as input $\Pi_{\mathcal{OT}}$ and outputs a one-sided simulatable OT. Our compiler is inspired by the work of [27]. In this the authors show how to combine k OTs (that we call OT candidates) to obtain an OT protocol that is secure against malicious sender even if $k-1$ of the OT candidates are insecure against malicious senders[15]. At a very high level the construction proposed in [27] works as follows. First Harnik et al. show a construction that works for $k = 2$ and then propose a generic compiler that transforms $(1,2)$-combiner into a $(1,k)$-combiner. The $(1,2)$-combiner works as follows. Consider two OT candidates $\Pi^0_{\mathcal{OT}}$ and $\Pi^1_{\mathcal{OT}}$. Let b be the input of the receiver and (l_0, l_1) be the input of the sender.

1. The sender chooses a random bit r
2. The receiver chooses random bits b_0, b_1 such that $b = b_0 \oplus b_1$.
3. The parties run $\Pi^0_{\mathcal{OT}}$ where the receiver uses b_0 as input and the sender uses the pair $(r, r \oplus l_0 \oplus l_1)$. The parties also run $\Pi^1_{\mathcal{OT}}$ where the receiver uses b_1 as input and sender uses $(r \oplus l_0, r \oplus l_1)$
4. The receiver output corresponds to the XOR of his outputs in both executions.

To extend the above construction to the case where $k > 2$, Harnik et al. consider k OT candidates and organize them as leaves of a binary tree, and applies the construction proposed above to every internal node (in a bottom up fashion). Now, by the properties of the combiner, for every node that securely implements OT, its ancestor must also securely implement OT. The output of the whole tree must therefore also securely implement OT since the root is an ancestor to all leaves. If the running time of the above $(1,2)$-combiner for malicious sender is m times that of its candidates, then the running time of the whole construction is $m^{\Omega(\log k)}$. Thus, in order for the running time to be polynomial, m must be a constant (which it is actually the case if we use the $(1,2)$-combiner showed in this section). We now denote with $\Pi_{\overline{\mathcal{OT}}} = (S_{\overline{\mathcal{OT}}}, R_{\overline{\mathcal{OT}}})$ the protocol obtained by combining $4\lambda^2$ parallel executions of $\Pi_{\mathcal{OT}}$ as described above, we prove that $\Pi_{\overline{\mathcal{OT}}}$ is secure with one-sided simulation accordingly to Def. 6.

In our formal description we assume, without loss of generality, that the sender's (receiver's) algorithm of $\Pi_{\mathcal{OT}}$ to compute its first message takes as input the security parameter, the input and a message (if any), and returns an auxiliary input and the first message to be sent. To compute the message for the round i, the sender's (receiver's) algorithm takes as input the auxiliary input and all the messages that have been send and received up to that round, and

[15] To prove our theorem we do not need a fully secure combiner. That is, we only need a combiner that guarantees security in the case that one execution of $\Pi_{\mathcal{OT}}$ is secure against malicious senders and all the executions of $\Pi_{\mathcal{OT}}$ are secure against malicious receivers.

returns the message to be send. We propose a formal description of $\Pi_{\overline{\mathcal{OT}}}$ in Fig. 3. To prove that $\Pi_{\overline{\mathcal{OT}}}$ is secure we cannot just rely on the security of the

Common input: Security parameters: $\lambda := 2^\kappa$ for some $k \in \mathbb{N}$, $n := 4\lambda^2$
Input to $R_{\overline{\mathcal{OT}}}$: $b \in \{0,1\}$. **Input to $S_{\overline{\mathcal{OT}}}$:** $l_0 \in \{0,1\}, l_1 \in \{0,1\}$.

$R_{\overline{\mathcal{OT}}} \rightarrow S_{\overline{\mathcal{OT}}}$
1. Run GB on input $(b, 1, \log(n))$ thus obtaining $b^1, \dots, b^n$.
2. For $i = 1, \dots, n$ run $R_{\mathcal{OT}}$ on input 1^λ and b^i thus obtaining $(\mathsf{aux}_r^i, \mathsf{ot}_1^i)$.
3. Send $\mathsf{ot}_1^1, \dots, \mathsf{ot}_1^n$ to $S_{\overline{\mathcal{OT}}}$

$S_{\overline{\mathcal{OT}}} \rightarrow R_{\overline{\mathcal{OT}}}$
1. Run GL on input $(l_0, l_1, i, \log(n))$ thus obtaining $(l_0^1, l_1^1), \dots, (l_0^n, l_1^n)$.
2. For $i = 1, \dots, n$ run $S_{\mathcal{OT}}$ on input 1^λ, ot_1^i, $(\mathsf{aux}_s^i, l_0^i, l_1^i)$ thus obtaining ot_2^i.
3. Send $\mathsf{ot}_2^1, \dots, \mathsf{ot}_2^n$ to $R_{\overline{\mathcal{OT}}}$.

$R_{\overline{\mathcal{OT}}} \rightarrow S_{\overline{\mathcal{OT}}}$
1. For $i = 1, \dots, n$ run $R_{\mathcal{OT}}$ on input $(\mathsf{ot}_1^i, \mathsf{ot}_2^i, \mathsf{aux}_r^i)$ thus obtaining ot_3^i.
2. Send $\mathsf{ot}_3^1, \dots, \mathsf{ot}_3^n$ to $S_{\overline{\mathcal{OT}}}$

$S_{\overline{\mathcal{OT}}} \rightarrow R_{\overline{\mathcal{OT}}}$
1. For $i = 1, \dots, n$ run $\Pi_{\mathcal{OT}}$ on input $(\mathsf{ot}_1^i, \mathsf{ot}_2^i, \mathsf{ot}_3^i, \mathsf{aux}_s^i)$ thus obtaining ot_4^i.
2. Send $\mathsf{ot}_4^1, \dots, \mathsf{ot}_4^n$ to $R_{\overline{\mathcal{OT}}}$.

Output Phase of $R_{\overline{\mathcal{OT}}}$

1. For $i = 1, \dots, n$ run $R_{\mathcal{OT}}$ on input $(\mathsf{ot}_1^i, \mathsf{ot}_2^i, \mathsf{ot}_3^i, \mathsf{ot}_4^i)$ and aux_r^i thus obtaining $l_{b^i}^i$.
2. Output $l_{b^1}^1 \oplus \cdots \oplus l_{b^n}^n$

```
GB(b, i, n)

    Pick $r \xleftarrow{\$} \{0,1\}$, compute $b_0 \leftarrow b \oplus r$ and set $b_1 \leftarrow r$.
    If $i = n$ then return $(b_0, b_1)$ else return $\mathsf{GB}(b_0, i+1, n)$, $\mathsf{GB}(b_1, i+1, n)$.

GL((l_0, l_1), i, n)

    Pick $r \xleftarrow{\$} \{0,1\}$, compute $l_{0,0} \leftarrow r$, $l_{0,1} \leftarrow r \oplus l_0 \oplus l_1$, $l_{1,0} \leftarrow r \oplus l_0$, $l_{1,1} \leftarrow r \oplus l_1$.
    If $i = n$ then return $(l_{0,0}, l_{0,1}), (l_{1,0}, l_{1,1})$
    else return $\mathsf{GL}((l_{0,0}, l_{0,1}), i+1, n)$, $\mathsf{GL}((l_{1,0}, l_{1,1}), i+1, n)$.
```

Fig. 3: Formal description of $\Pi_{\overline{\mathcal{OT}}}$

combiner since a malicious sender could sample the trapdoor functions in such a way that the security of all the OT executions is compromised. We show that this can happen only with negligible probability. We denote with $\Pi_{\mathcal{OT}}^i$ the i-th execution of $\Pi_{\mathcal{OT}}$ in a run of $\Pi_{\overline{\mathcal{OT}}}$. To denote the messages of $\Pi^i_{\mathcal{OT}}$ we extend the notation used in the description of $\Pi_{\mathcal{OT}}$ by writing m^i (or m_i) if m is a symbol used in the description of $\Pi_{\mathcal{OT}}$ (e.g., in the second round of $\Pi^i_{\mathcal{OT}}$ the sender sends $f^i_{0,0}, \dots f^i_{1,1}, \beta^i, R^i_0, R^i_1$). At a high level the proof works in this way. If by contradiction all the OT executions are insecure this implies that in any of the OT executions the malicious sender sends the TDPs $(f^i_{0,0}, f^i_{0,1}, f^i_{1,0}, f^i_{1,1})$ such that for all $p_i \in \{0,1\}$

1. if the instance $(f^i_{p_i,0}, f^i_{p_i,1})$ is used to run the DWE scheme then hiding of the DWE would not hold and
2. if $(f^i_{1-p_i,0}, f^i_{1-p_i,1})$ are used to run the remaining computation of $\Pi^i_{\mathcal{OT}}$ then $\Pi^i_{\mathcal{OT}}$ would be insecure (i.e., $(f^i_{1-p_i,0}, f^i_{1-p_i,1})$ might not be injective).

This means that any OT executions $\Pi^i_{\mathcal{OT}}$ has a pair of TDPs $(f^i_{\mathsf{d}',0}, f^i_{\mathsf{d}',1})$ with $\mathsf{d}' \in \{0,1\}$ that are not injective and that have a collision set smaller than $2^{-1}|D_\alpha|$. However, we note that if $\mathsf{d}_i = \mathsf{d}'_i$ in a sufficiently large number of executions then we have that the there is an execution j where $r^j_0 \oplus R^j_0$ and $r^j_1 \oplus R^j_1$ are such that $y^j_0 \leftarrow S(f^j_{d_j,0}, r^j_0 \oplus R^j_0)$ and $y^j_1 \leftarrow S(f^j_{d_j,1}, r^j_1 \oplus R^j_1)$ have exactly one pre-image each with overwhelming probability. This would allow us to apply the lemma 1 and state that $\Pi^i_{\mathcal{OT}}$ is secure. Then we can simply rely on the security of the combiner to claim that $\Pi_{\overline{\mathcal{OT}}}$ is secure. To argue that such a value j exists we use the fact that the receiver picks d_i randomly in $\{0,1\}$ for all $i \in \{1, \dots, 4\lambda^2\}$.

Theorem 8. *If enhanced permutations with efficiently sampleable range and domain exist, then $\Pi_{\overline{\mathcal{OT}}}$ securely realizes the oblivious transfer functionality $F_{\mathcal{OT}}$ with one-sided simulation with black-box use of the underlying primitive.*

We refer to the full version for the proof of the theorem. The protocol $\Pi_{\overline{\mathcal{OT}}}$ described in this section restricts the sender to use two bits as input (bit-OT). In some applications (as the one that we are going to consider in this work) it is crucial that the sender input is represented by strings $l_0 \in \{0,1\}^\kappa, l_1 \in \{0,1\}^\kappa$ with $\kappa \in \mathbb{N}$ (string-OT). The work of Brassard et al. [4] proposes a way to construct an information theoretically secure string OT protocol from an information theoretically secure bit OT protocol. The idea proposed in [4] is to use run κ bit-OT executions in such a way that that regardless of the choices of the input bits of malicious receivers in these executions, he can only obtain one of the two inputs. We show how to use the technique proposed in [4] to transform our bit-OT protocol $\Pi_{\overline{\mathcal{OT}}}$ into a string-OT protocol $\Pi^\kappa_{\overline{\mathcal{OT}}} := (S^\kappa_{\overline{\mathcal{OT}}}, R^\kappa_{\overline{\mathcal{OT}}})$. We refer the reader to the full version for the formal description of the protocol and its proof. We note that $\Pi^\kappa_{\overline{\mathcal{OT}}}$ can be easily run in parallel polynomialy many times.

8 Black-Box Round Optimal 2PC.

In [35, Sec. 3.2] the authors show how to obtain a fully simulatable OT protocol using in a black-box way: (parallel) one-sided simulatable OTs and one-to-one one-way functions. Using this result we can state the following theorem.

Theorem 9. *If enhanced trapdoor permutations with efficiently sampleable range and domain exist, then there exists a 4-round protocol $\mathcal{OT}$ that securely realizes the oblivious transfer functionality $F^m_{\mathcal{OT}}$ with black-box use of the underlying primitive.*

An immediate corollary from the above result, in conjunction with the work of [28] building a non-interactive secure two-party protocol in the OT-hybrid model is the following.

Corollary 2. *If enhanced trapdoor permutations with efficiently sampleable range/ domain and one-to-one OWFs exist, then there exists a round optimal protocol that securely realizes any 2-party functionality with BB use of the primitives.*

Acknowledgments

Arka Rai Choudhuri is supported by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, NSF CNS-1908181, Office of Naval Research N00014-19-1-2294.

Michele Ciampi has done part of this work while consulting for Stealth Software Technologies, Inc.

Vipul Goyal is supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

Abhishek Jain is supported in part by an NSF CNS grant 1814919, NSF CAREER award 1942789, Johns Hopkins University Catalyst award and Office of Naval Research grant N00014-19-1-2294.

Rafail Ostrovsky is supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, NSF grant CNS-2001096, US-Israel BSF grant 2015782, Google Faculty Award, JP Morgan Faculty Award, IBM Faculty Research Award, Xerox Faculty Research Award, OKAWA Foundation Research Award, B. John Garrick Foundation Award, Teradata Research Award, Lockheed-Martin Research Award and Sunday Group. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

References

1. Bellare, M., Yung, M.: Certifying cryptographic tools: The case of trapdoor permutations. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 442–460. Springer, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_31
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC. pp. 1–10. ACM Press (May 1988). https://doi.org/10.1145/62212.62213
3. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_17
4. Brassard, G., Crépeau, C., Santha, M.: Oblivious transfers and intersecting codes. IEEE Trans. Information Theory **42**(6), 1769–1780 (1996). https://doi.org/10.1109/18.556673, https://doi.org/10.1109/18.556673
5. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press (Jun 2019). https://doi.org/10.1145/3313276.3316380
6. Canetti, R., Lichtenberg, A.: Certifying trapdoor permutations, revisited. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 476–506. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_18
7. Chailloux, A., Ciocan, D.F., Kerenidis, I., Vadhan, S.P.: Interactive and noninteractive zero knowledge are equivalent in the help model. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 501–534. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_28
8. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: 20th ACM STOC. pp. 11–19. ACM Press (May 1988). https://doi.org/10.1145/62212.62214
9. Choudhuri, A.R., Ciampi, M., Goyal, V., Jain, A., Ostrovsky, R.: Round optimal secure multiparty computation from minimal assumptions. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 291–319. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_11
10. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Round-optimal secure two-party computation from trapdoor permutations. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 678–710. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_23
11. Ciampi, M., Parisella, R., Venturi, D.: On adaptive security of delayed-input sigma protocols and fiat-shamir NIZKs. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 670–690. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_33
12. Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.: On the communication required for unconditionally secure multiplication. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 459–488. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_16
13. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO'82. pp. 205–210. Plenum Press, New York, USA (1982)

14. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS. pp. 308–317. IEEE Computer Society Press (Oct 1990). https://doi.org/10.1109/FSCS.1990.89549
15. Friolo, D., Masny, D., Venturi, D.: A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. TCC 2019 (2019), https://eprint.iacr.org/2018/473
16. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 448–476. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_16
17. Garg, S., Ostrovsky, R., Visconti, I., Wadia, A.: Resettable statistical zero knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 494–511. Springer, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9_28
18. Goldberg, S., Reyzin, L., Sagga, O., Baldimtsi, F.: Efficient noninteractive certification of RSA moduli and beyond. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 700–727. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_24
19. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge, UK (2004)
20. Goldreich, O.: Computational complexity: a conceptual perspective. SIGACT News **39**(3), 35–39 (2008). https://doi.org/10.1145/1412700.1412710, https://doi.org/10.1145/1412700.1412710
21. Goldreich, O.: Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art (2011)
22. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology **7**(1), 1–32 (Dec 1994). https://doi.org/10.1007/BF00195207
23. Goldreich, O., Rothblum, R.D.: Enhancements of trapdoor permutations. Journal of Cryptology **26**(3), 484–512 (Jul 2013). https://doi.org/10.1007/s00145-012-9131-8
24. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985). https://doi.org/10.1145/22145.22178
25. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). https://doi.org/10.1137/0218012, https://doi.org/10.1137/0218012
26. Haitner, I.: Implementing oblivious transfer using collection of dense trapdoor permutations. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 394–409. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_22
27. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_6
28. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_23
29. Kakvi, S.A., Kiltz, E., May, A.: Certifying RSA. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 404–414. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_25

30. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_21
31. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC. pp. 20–31. ACM Press (May 1988). https://doi.org/10.1145/62212.62215
32. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press (May 1992). https://doi.org/10.1145/129712.129782
33. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 171–189. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_10
34. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM (Jan 2001)
35. Ostrovsky, R., Richelson, S., Scafuro, A.: Round-optimal black-box two-party computation. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 339–358. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_17
36. Ostrovsky, R., Venkatesan, R., Yung, M.: Fair games against an all-powerful adversary. In: Cai, J. (ed.) Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 155–169. DIMACS/AMS (1990). https://doi.org/10.1090/dimacs/013/09, https://doi.org/10.1090/dimacs/013/09
37. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_31
38. Rabin, M.O.: Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (Jan 1979)
39. Rothblum, R.: A taxonomy of enhanced trapdoor permutations. Electronic Colloquium on Computational Complexity (ECCC) **17**, 145 (2010), http://eccc.hpi-web.de/report/2010/145
40. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982). https://doi.org/10.1109/SFCS.1982.38
41. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25