Oblivious Transfer from Trapdoor Permutations in Minimal Rounds

Arka Rai Choudhuri* Michele Ciampi† Vipul Goyal‡ Abhishek Jain§
Rafail Ostrovsky¶

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.

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) [Kil88]. In fact, recent works [BL18, CCG+20] 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

*Johns Hopkins University. achoud@cs.jhu.edu
†The University of Edinburgh. michele.ciampi@ed.ac.uk
‡Carnegie Mellon University and NTT Research. goyal@cs.cmu.edu
§Johns Hopkins University. abhishek@cs.jhu.edu
¶University of California, Los Angeles. rafail@cs.ucla.edu

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[1] 1-out-of-2 OT protocols [EGL82] were based on TDPs. Subsequent works devised compilation strategies to transform the protocol of [EGL82] to the setting of malicious senders and receivers. In particular, [KO04] constructed a four-round OT protocol that makes non-black-box use of TDPs. More recently, [ORS15] 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) [KO04].

A significant disadvantage of these works (including [EGL82]), 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) [GRSB19, FLS90, BY93, Gol04, Gol08, Gol11, GR13, CL18, KKM12]. 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 [OVY90], 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.

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 [FLS90]. 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, [BY93] 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 [Gol08, Gol11, Gol04, GR13]

[1] 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.

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 2.1). [CL18] 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 [GRSB19] 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 $(f(k), k \oplus m)$[2] 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

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

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 [GMR85][3].

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)* [CCKV08] 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 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 [GOVW12] 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 [IKO+11] 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 [ORS15, KO04] 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

[3] 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.

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.

1.2 Organization of the Paper

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

2 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, $x \xleftarrow{\$} Q$ 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 Appendix A.2 and Appendix A.3 for the formal definitions.

2.1 Injective TDFs and TDPs

In this section we define the notion of trapdoor function following mostly the notation proposed in [CL18].

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* 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:*

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

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

$$\text{Prob}\left[\ (\alpha \xleftarrow{\$} I_0(1^\lambda); y \xleftarrow{\$} S_R(\alpha); x' \xleftarrow{\$} \mathcal{A}(\alpha, y) : \mathsf{Eval}(\alpha, x') = y\ \right] \leq \nu(\lambda). \tag{2}$$

Additionally, it is required that, for any $\alpha \xleftarrow{\$} I_0(1^\lambda)$, the distribution sampled by S_R should be close to 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 $\alpha \xleftarrow{\$} I_0(1^\lambda)$, 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 $\alpha \xleftarrow{\$} I_0(1^\lambda)$, the resulting primitive is a trapdoor permutation. If for any $\alpha \xleftarrow{\$} I(1^\lambda)$, $S_D = \{0,1\}^{\mathsf{poly}(\lambda)}$, 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*

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

2.1.1 Enhancements

Goldreich [Gol04] 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 [CL18] 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 [Gol04])**.** *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*

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

Definition 4 (Enhanced Hard-Core Predicate [Gol11]). *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*

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

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

$$\{(\alpha, r, h(\alpha, \mathsf{Inv}(\mathtt{td}, S_R(\alpha, r)))) : (\alpha, \mathtt{td}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), r \xleftarrow{\$} \{0,1\}^\star\}$$

$$\{(\alpha, r, u) : \alpha \xleftarrow{\$} I_0(1^\lambda), r \xleftarrow{\$} \{0,1\}^\star, u \xleftarrow{\$} \{0,1\}\}$$

2.1.2 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 $r \xleftarrow{\$} \{0,1\}^\lambda$, $S(\alpha; r)$ 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 [Rab79]). 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.

2.2 Commit-and-Open Protocols

In [FMV19] 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 [FMV19], 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. [FMV19])**.** *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*

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

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*

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

The authors of [FMV19] show that one of the protocols proposed in [ORS15] 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 [FMV19] for more discussion on the notion of commit-and-open and for its black-box instantiation from one-to-one one-way-functions.

2.3 Oblivious Transfer and 2-PC

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

Definition 6 ([ORS15])**.** *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 ([ORS15])**.** *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.

2.3.1 Secure Two-Party Computation [ORS15]

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 $P_i^\star$.

3 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* PK-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:*

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

3.1 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 [CPV20, CCH+19] on these sigma protocols. Following [CPV20], 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 [CPV20] the authors observe that the above protocol has the following interesting property. There exists a PPT algorithm 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 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 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 ChallExt as the *bad-challenge extractor*. The same holds true for

the classical Sigma protocol for QR [GMR89] (along the lines of the full version of [CCH+19, 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* $\mathcal{R}_{L_1}$.

3.1.1 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).

4 Black-Box DWE for Trapdoor Permutations

A function $f_\alpha : D_\alpha \to D_\alpha$ 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* $f_\alpha : \{D_\alpha \to D_\alpha\}$. *The* collision set *of* f_α, *denoted with* $C(f_\alpha)$, *is* $\{x_1 \in D_\alpha : \exists x_2 \in D_\alpha \text{ s.t. } x_1 \neq x_2 \text{ and } \mathsf{Eval}(\alpha, x_1) = \mathsf{Eval}(\alpha, x_2)\}$. *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 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).

4.1 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 $y_i \leftarrow f_\alpha(x_i)$.[4]
  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 $(j_i, y_i, c_i)_{i \in [n]}$.
  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 App. D.3 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 $(\mathsf{Enc}_1^{\mathsf{f}}, \mathsf{Dec}_1^{\mathsf{f}})$.

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

4.1.1 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 $L_0^{\mathsf{2f}} := \{\alpha_0, \alpha_1 : |C(f_{\alpha_0})| \geq 2^{-1}|D_{\alpha_0}| \text{ or } |C(f_{\alpha_1})| \geq 2^{-1}|D_{\alpha_1}|\}$ 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.

5 *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 [ORS15]. 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[5] 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.

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.

[5] 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.

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 6 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 $\mathcal{F} := (\mathsf{Gen}, S, S_{DR}, f, f^{-1})$[6] 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. 4.

We now give an informal description of our protocol and refer to Fig. 1 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 V_1. If it is not, then $S_{\mathcal{OT}}$ stops and outputs $\perp$, otherwise she computes $\omega_0 \leftarrow f_{\mathsf{d},0}^{-1}(S(f_{\mathsf{d},0}, r_0 \oplus R_0))$ and $\omega_1 \leftarrow f_{\mathsf{d},1}^{-1}(S(f_{\mathsf{d},1}, r_1 \oplus R_1))$. 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 $l_b = W_b^1 \oplus h(f_{\mathsf{d},b}, z_1')$.

In Fig. 1 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* $\{\mathsf{REAL}_{\Pi_{\mathcal{OT}}, R^\star_{\mathcal{OT}}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda} \approx \mathsf{IDEAL}_{F_{\mathcal{OT}}, \mathsf{Sim}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda}$.[7]

We refer to App. D.1 for the formal proof of the theorem.

[6] 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}.

[7] We refer to Sec. 2.3 for a formal definition of $\mathsf{REAL}_{\Pi_{\mathcal{OT}}, R^\star_{\mathcal{OT}}(z)}$ and $\mathsf{IDEAL}_{F_{\mathcal{OT}}, \mathsf{Sim}(z)}$

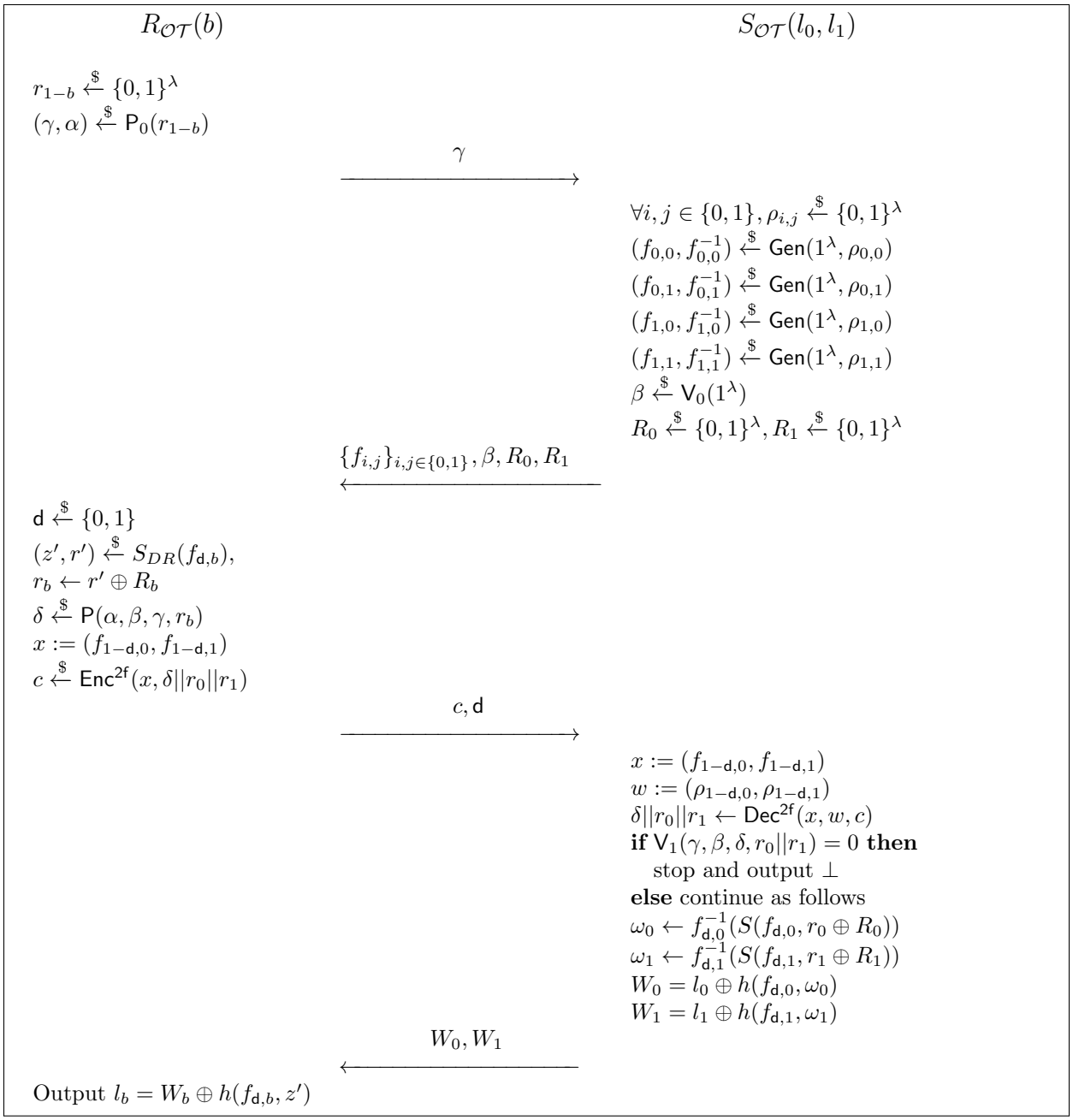

Figure 1: Description of $\Pi_{\mathcal{OT}}$.

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

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 App. D.2 for the formal proof of the theorem.

We now prove a lemma that will be helpful hereafter. Before doing that, 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. $\text{Prob}\left[\ \mathsf{E_g}^{i,j}\ \right] \geq 1 - \nu(\lambda)\ \forall i, j \in \{0,1\}$ *or*
2. $\text{Prob}\left[\ \mathsf{E_g}^{0,0}\ \right] < 2^{-1}$ *or* $\text{Prob}\left[\ \mathsf{E_g}^{0,1}\ \right] < 2^{-1}$ *and* $\text{Prob}\left[\ \mathsf{E_g}^{1,0}\ \right] < 2^{-1}$ *or* $\text{Prob}\left[\ \mathsf{E_g}^{1,1}\ \right] < 2^{-1}$

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

6 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
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 [HKN+05]. 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[8]. At a very high level the construction proposed in [HKN+05] 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 $(r \oplus l_0, r \oplus l_1)$

[8] 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.

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 returns the message to be send. We propose a formal description of $\Pi_{\overline{\mathcal{OT}}}$ in Fig. 2. To prove that $\Pi_{\overline{\mathcal{OT}}}$ is secure we cannot just rely on the security of the 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 $\Pi^i_{\mathcal{OT}}$ 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 App. D.2.1 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

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 $b^1, \ldots, b^n$.
2. For $i = 1, \ldots, n$ run $R_{\mathcal{OT}}$ on input 1^λ and b^i thus obtaining $(\mathsf{aux}_r^i, \mathsf{ot}_1^i)$.
3. Send $\mathsf{ot}_1^1, \ldots, \mathsf{ot}_1^n$ 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 $(l_0^1, l_1^1), \ldots, (l_0^n, l_1^n)$.
2. For $i = 1, \ldots, n$ 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 $\mathsf{ot}_2^1, \ldots, \mathsf{ot}_2^n$ to $R_{\overline{\mathcal{OT}}}$.

$R_{\overline{\mathcal{OT}}} \rightarrow S_{\overline{\mathcal{OT}}}$

1. For $i = 1, \ldots, n$ 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 $\mathsf{ot}_3^1, \ldots, \mathsf{ot}_3^n$ to $S_{\overline{\mathcal{OT}}}$

$S_{\overline{\mathcal{OT}}} \rightarrow R_{\overline{\mathcal{OT}}}$

1. For $i = 1, \ldots, n$ 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 $\mathsf{ot}_4^1, \ldots, \mathsf{ot}_4^n$ to $R_{\overline{\mathcal{OT}}}$.

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

1. For $i = 1, \ldots, n$ 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 $l_{b^1}^1 \oplus \cdots \oplus l_{b^n}^n$

$\mathsf{GB}(b, i, n)$

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 $\mathsf{GB}(b_0, i+1, n)$, $\mathsf{GB}(b_1, i+1, n)$.

$\mathsf{GL}((l_0, l_1), i, n)$

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 $\mathsf{GL}((l_{0,0}, l_{0,1}), i+1, n)$, $\mathsf{GL}((l_{1,0}, l_{1,1}), i+1, n)$.

Figure 2: Formal description of $\Pi_{\overline{\mathcal{OT}}}$

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. [BCS96] proposes a way to construct an information theoretically secure string OT protocol from an information theoretically secure bit OT protocol. The idea proposed in [BCS96] 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 [BCS96] 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 App. C 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. The proof of this claim follows arguments similar to the arguments proposed in the proof of the Theorem 13.

7 Black-Box Round Optimal 2PC

In [ORS15, 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 $F_{\mathcal{OT}}^{m}$ with black-box use of the underlying primitive.*

An immediate corollary from the above result, in conjunction with the work of [IKO+11] 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

[BCS96] Gilles Brassard, Claude Crépeau, and Miklos Santha. Oblivious transfers and intersecting codes. *IEEE Trans. Information Theory*, 42(6):1769–1780, 1996.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BY93] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 442–460, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 291–319, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 1082–1090, Phoenix, AZ, USA, June 23–26, 2019. ACM Press.

[CCKV08] André Chailloux, Dragos Florin Ciocan, Iordanis Kerenidis, and Salil P. Vadhan. Interactive and noninteractive zero knowledge are equivalent in the help model. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 501–534, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 476–506, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[CPV20] Michele Ciampi, Roberto Parisella, and Daniele Venturi. On adaptive security of delayed-input sigma protocols and fiat-shamir nizks. *SCN*, 2020:831, 2020.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium*

on Foundations of Computer Science, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[FMV19] Daniele Friolo, Daniel Masny, and Daniele Venturi. 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.

[GKOV12] Sanjam Garg, Abishek Kumarasubramanian, Rafail Ostrovsky, and Ivan Visconti. Impossibility results for static input secure computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 424–442, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *SIGACT News*, 39(3):35–39, 2008.

[Gol11] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. 2011.

[GOVW12] Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 494–511, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *Journal of Cryptology*, 26(3):484–512, July 2013.

[GRSB19] Sharon Goldberg, Leonid Reyzin, Omar Sagga, and Foteini Baldimtsi. Efficient noninteractive certification of RSA moduli and beyond. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 700–727, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[KKM12] Saqib A. Kakvi, Eike Kiltz, and Alexander May. Certifying RSA. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 404–414, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

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

[OVY90] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Fair games against an all-powerful adversary. In Jin-Yi Cai, editor, *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 155–169. DIMACS/AMS, 1990.

[Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.

A Additional notions

A.1 Parallel OT

Definition 10 (Parallel oblivious transfer functionality $F_{\mathcal{OT}}^m$ [ORS15])**.** *The parallel Oblivious Transfer Functionality $F_{\mathcal{OT}}^m$ is identical to the functionality $F_{\mathcal{OT}}$, with the difference that takes in input m pairs of string from S $(l_0^1, l_1^1, \ldots, l_0^m, l_1^m)$ (whereas $F_{\mathcal{OT}}$ takes just one pair of strings from S) and m bits from R, $b_1, \ldots, b_m$ (whereas $F_{\mathcal{OT}}$ takes one bit from R) and outputs to the receiver values $(l_{b_1}^1, \ldots, l_{b_m}^m)$ while the sender receives nothing.*

Definition 11 ([ORS15])**.** *Let $F_{\mathcal{OT}}^m$ be the Oblivious Transfer functionality as described in Def. 10. We say that a protocol Π securely realizes $F_{\mathcal{OT}}^m$ with one-sided simulation if the following holds:*

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

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

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

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

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

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

A.2 Standard Notions

Definition 12 (Computational indistinguishability)**.** *Let* $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *be ensembles, where* X_λ*'s and* Y_λ*'s are probability distribution over* $\{0,1\}^l$*, for same* $l = \mathsf{poly}(\lambda)$*. We say that* $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *are* computationally indistinguishable*, denoted* $X \approx Y$*, if for every* PPT *distinguisher* $\mathcal{D}$ *there exists a negligible function* ν *such that for sufficiently large* $\lambda \in \mathbb{N}$*,*

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

We note that in the usual case where $|X_\lambda| = \Omega(\lambda)$ and λ can be derived from a sample of X_λ, it is possible to omit the auxiliary input 1^λ. In this paper we also use the definition of *Statistical Indistinguishability*. This definition is the same as Definition 12 with the only difference that the distinguisher $\mathcal{D}$ is unbounded. In this case use $X \equiv Y$ to denote that two ensembles are statistically indistinguishable.

A.3 Sigma Protocols and the DDH assumption

Let L be an NP language, with corresponding relation $\mathcal{R}$. A Sigma protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ is a 3-round public-coin protocol. In particular, an execution of Σ proceeds as follows:

- The prover $\mathcal{P}$ computes the first message using as input the instance to be proved $x \in L$ with the corresponding witness w, and outputs the first message a with an auxiliary information $\mathtt{aux}$; we denote this action with $(a, \mathtt{aux}) \leftarrow\!\!\$\ \mathcal{P}(x, w)$.

- The verifier $\mathcal{V}$, upon receiving a, sends a random string $c \leftarrow\!\!\$\ \{0,1\}^\ell$ with $\ell \in \mathbb{N}$.
- The prover $\mathcal{P}$, upon input c and $\mathtt{aux}$, computes and sends z to $\mathcal{V}$; we denote this action with $z \leftarrow\!\!\$\ \mathcal{P}(\mathtt{aux}, c)$.
- The verifier $\mathcal{V}$, upon input (x, a, c, z), outputs 1 to accept and 0 to reject; we denote this action with $\mathcal{V}(x, a, c, z) = d$ where $d \in \{0,1\}$ denotes whether $\mathcal{V}$ accepts or not.

Definition 13 (Sigma protocol [CDS94])**.** *A 3-move protocol Σ with challenge length $\ell \in \mathbb{N}$ is a* Sigma protocol *for a relation $\mathcal{R}$ if it enjoys the following properties:*

- ***Completeness.*** *If $(x, w) \in \mathcal{R}$, then all honest 3-move transcripts for (x, w) are accepting.*
- ***Special soundness.*** *There exists an efficient algorithm* Extract *that, on input two accepting transcripts (a, c, z) and (a, c', z') for x with $c' \neq c$ (we refer to such two accepting transcripts as a* collision*) outputs a witness w such that $(x, w) \in \mathcal{R}$.*
- ***Special honest-verifier zero knowledge (SHVZK).*** *There exists a* PPT *simulator algorithm* Sim *that takes as input $x \in L$ and $c \in \{0,1\}^\ell$, and outputs an accepting transcript for x where c is the challenge (we denote this action with $(a, z) \leftarrow\!\!\$\ \mathsf{Sim}(x, c)$). Moreover, for all ℓ-bit strings c, the distribution of the output of the simulator on input (x, c) is identical to the distribution of the 3-move honest transcript obtained when $\mathcal{V}$ sends c as challenge and $\mathcal{P}$ runs on common input x and any private input w such that $(x, w) \in \mathcal{R}$.*

A.3.1 The DDH Assumption.

Let $\mathcal{G}$ be a cyclic group with generator g, and let A, B and X be elements of $\mathcal{G}$. We say that (g, A, B, X) is a *Diffie-Hellman tuple* (a *DH tuple*, in short) if $A = g^\alpha, B = g^\beta$ for some integers $0 \leq \alpha, \beta \leq |\mathcal{G}| - 1$ and $X = g^{\alpha\beta}$. If this is not the case, the tuple is called *non-DH*. To verify that a tuple is DH, it is sufficient to have the discrete log α of A to the base g, and then to check that $X = B^\alpha$. We thus define the polynomial-time relation $\mathcal{R}_{DH} = \{((g, A, B, X), \alpha) : A = g^\alpha \text{ and } X = B^\alpha\}$ of all DH tuples.

The *Decisional Diffie-Hellman* assumption (the DDH assumption) posits the hardness of distinguishing a randomly selected DH tuple from a randomly selected non-DH tuple with respect to a *group generator* algorithm. For sake of concreteness, we consider the specific group generator GG that, on input 1^λ, randomly selects a λ-bit prime p such that $q = (p-1)/2$ is also prime and outputs the (description of the) order q group $\mathcal{G}$ of the quadratic residues modulo p along with a random generator g of $\mathcal{G}$.

Assumption 1 (DDH Assumption)**.** *For every PPT algorithm $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ s.t.*

$$\Big|\text{Prob}\Big[\,\mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^\gamma) = 1 : (\mathcal{G}, q, g) \xleftarrow{\$} \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_q\,\Big] -$$
$$\text{Prob}\Big[\,\mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^{\alpha\beta}) = 1 : (\mathcal{G}, q, g) \xleftarrow{\$} \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_q\,\Big]\Big| \leq \nu(\lambda).$$

B 3-Round OT Secure Against Malicious Senders

B.1 *Almost* Secure 3-round OT

In this section we propose a 3-round protocol $\Pi_{\mathcal{OT}}^{\mathsf{sh-R}} := (S_{\mathcal{OT}}^{\mathsf{sh-R}}, R_{\mathcal{OT}}^{\mathsf{sh-R}})$ that is secure against semi-honest receiver and that retains the privacy of the receiver's input against malicious senders under some conditions. In more details, we show that the privacy of the receiver's input is preserved as long as the sender chooses the trapdoor permutation accordingly to a predefined distribution. In the next section we show how to modify $\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}$ to make it resilient against any adversarial behavior.

To construct $\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}$ we make use the following tools.

1. An enhanced trapdoor permutation with efficiently sampleable range and domain $\mathcal{F} := (\mathsf{Gen}, S, S_{DR}, f, f^{-1})$
2. The DWE scheme $(\mathsf{Enc}^{\mathsf{f}}, \mathsf{Dec}^{\mathsf{f}})$ for the languages (L_0, L_1) where $L_0 = \{\alpha : |C(f_\alpha)| \geq 2^{-1}|D_\alpha|\}$ and $L_1 = \{\alpha : (\alpha, \mathtt{td}) \leftarrow \mathsf{Gen}(1^\lambda; r), r \in \{0,1\}^\lambda\}$.

We now give an informal description of our protocol and refer to Fig. 3 for the formal description. Let $b \in \{0,1\}$ be the input of $R_{\mathcal{OT}}^{\mathsf{sh-R}}$ and $l_0, l_1 \in \{0,1\}$ be the input of $S_{\mathcal{OT}}^{\mathsf{sh-R}}$.

In the **first round** $S_{\mathcal{OT}}^{\mathsf{sh-R}}$ picks a random string $\rho \xleftarrow{\$} \{0,1\}$, generates a trapdoor permutation $(f, f^{-1}) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{f}}(1^\lambda; \rho)$, and sends f to $R_{\mathcal{OT}}^{\mathsf{sh-R}}$.

In the **second round** $R_{\mathcal{OT}}^{\mathsf{sh-R}}$ computes $(z', z_b) \xleftarrow{\$} S_{DR}(f)$ and $z_{1-b} \xleftarrow{\$} S_D(f)$. Then $R_{\mathcal{OT}}^{\mathsf{sh-R}}$ encrypts these values using the DWE scheme on input the public-key $\mathsf{pk} := f$ thus obtaining c and sends c to $S_{\mathcal{OT}}^{\mathsf{sh-R}}$.

In the **third round** $S_{\mathcal{OT}}^{\mathsf{sh-R}}$ decrypts c using the witness ρ thus obtaining z_0, z_1. Then $S_{\mathcal{OT}}^{\mathsf{sh-R}}$ computes $\omega_0 \leftarrow f^{-1}(z_0)$ and $\omega_1 \leftarrow f^{-1}(z_1)$. Then for $j = 0, 1$, $S_{\mathcal{OT}}^{\mathsf{sh-R}}$ encrypts the input l_j via one-time pad using as a key the output of the hard-core predicate of f on input ω_j thus obtaining W_j. $S_{\mathcal{OT}}^{\mathsf{sh-R}}$ then sends (W_0, W_1) to $R_{\mathcal{OT}}^{\mathsf{sh-R}}$ and stops.

In the **output phase**, $R_{\mathcal{OT}}^{\mathsf{sh-R}}$ computes and outputs $l_b = W_b \oplus h(f, z')$.

In Fig. 3 we propose a formal description of the protocol.

Theorem 10. *For every non-uniform adversary $S_{\mathcal{OT}}^{\mathsf{sh-R}^\star}$ controlling the sender, if one of the following holds with overwhelming probability*

1. *f is almost a permutation or*
2. *$f \in L_0$.*

then the two distributions are statistically indistinguishable

$$\{\mathsf{View}^{R_{\mathcal{OT}}^{\mathsf{sh-R}}}_{\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}, S^\star(z)}(l_0, l_1, 0)\}_{z \in \{0,1\}^\star}, \{\mathsf{View}^{R_{\mathcal{OT}}^{\mathsf{sh-R}}}_{\Pi_{\mathcal{OT}}, S^\star(z)}(l_0, l_1, 1)\}_{z \in \{0,1\}^\star}$$

where $\mathsf{View}^{R_{\mathcal{OT}}}_{\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}, S^\star(z)}$ denotes the view of adversary $S^\star$ after a real execution of protocol $\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}$ with the honest receiver $R_{\mathcal{OT}}^{\mathsf{sh-R}}$.

Proof. We divide the proof is two parts, one for each of the case listed above.

Case 1. Note that in this case we have the guarantee that the functions f is almost a permutation, therefore we can just rely on the fact that f is an almost permutations with efficiently samplable domain and range.

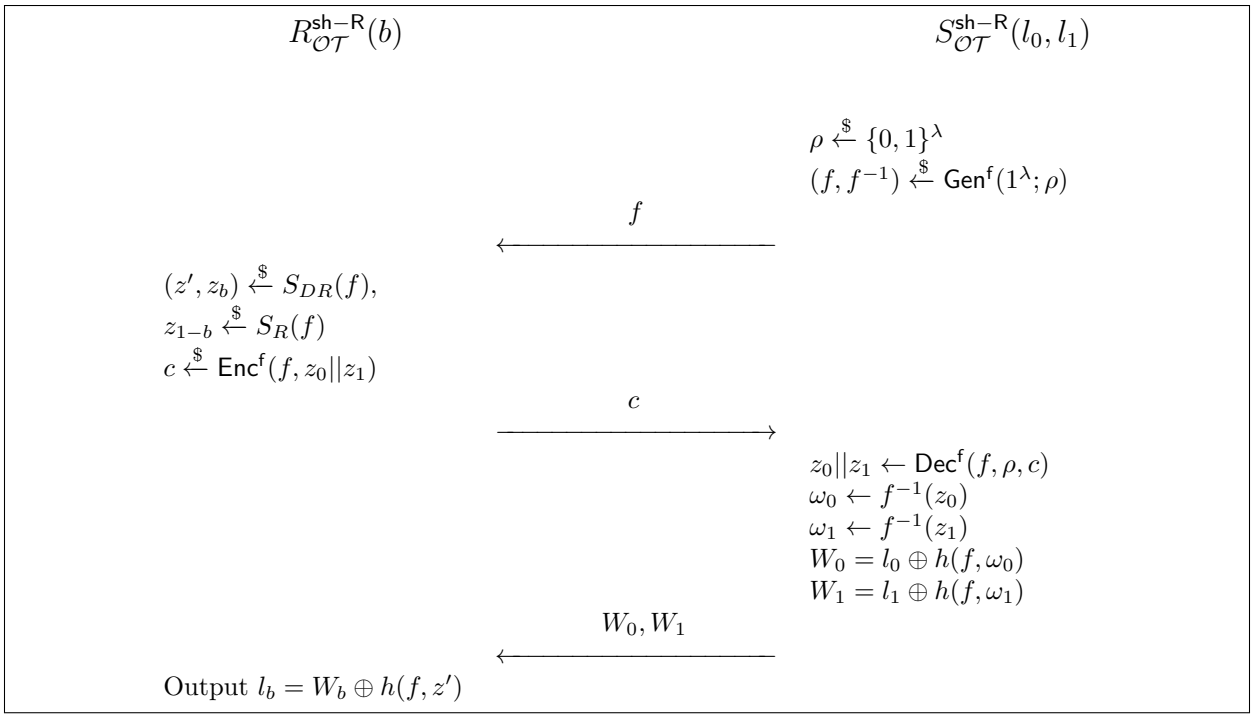

Figure 3: Description of $\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}$.

Case 2. In this case we can rely on the hiding of $(\mathsf{Enc}^{\mathsf{f}}, \mathsf{Dec}^{\mathsf{f}})$ since $f \in L_0$. Therefore, the encryption sent in the third round hides in an information theoretical sense all the values computed by $R^{\mathsf{sh-R}}_{\mathcal{OT}}$ that depends on the function f (which might not be injective and thus leaking the bit of the receiver).

The prof proceeds similarly to the proof for the case 1 with the following differences. We consider the hybrid $H^{\perp}_0$ that is exactly like $\mathsf{View}^{R^{\mathsf{sh-R}}_{\mathcal{OT}}}_{\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}, S^\star(z)}(l_0, l_1, 0)$ except for the fact that c contains and encryption of 0^λ. $\{\mathsf{View}^{R^{\mathsf{sh-R}}_{\mathcal{OT}}}_{\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}, S^\star(z)}(l_0, l_1, 0)\} \equiv H^{\perp}_0$ due to the hiding of the DWE scheme. Similarly, we consider the hybrid $H^{\perp}_1$ that is exactly like $\mathsf{View}^{R^{\mathsf{sh-R}}_{\mathcal{OT}}}_{\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}, S^\star(z)}(l_0, l_1, 1)$ except for the fact that c contains an encryption of 0^λ.

To prove that $H^{\perp}_0 \equiv \mathsf{View}^{R^{\mathsf{sh-R}}_{\mathcal{OT}}}_{\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}, S^\star(z)}(l_0, l_1, 0)$ we just rely on the fact that f has efficiently samplable domain and range algorithms.

□

Theorem 11. *If enhanced trapdoor permutations exists, the $\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}$ is secure against semi-honest $R^{\mathsf{sh-R}}_{\mathcal{OT}}$.*

Proof. The proof of this theorem follows almost the same argument of [GR13, Claim 3.1] The simulator on input (b, l_b), sample three strings $r_1, r_2, r_3 \xleftarrow{\$} \{0,1\}^\lambda$ and does the following steps.

1. $f \xleftarrow{\$} \mathsf{Gen}^{\mathsf{f}}(1^\lambda)$.

2. $(z', z_b) \leftarrow S_{DR}(f; r_1)$, $z_{1-b} \leftarrow S_D(f; r_2)$.

3. $c \leftarrow \mathsf{Enc}^{\mathsf{f}}(f, z_0||z_1; r_3)$.

4. $W_0 \leftarrow l_b \oplus h(f, z')$, $W_1 \leftarrow \{0,1\}^{\lambda}$.

5. Output $(b, (r_1, r_2, r_3), f, c, W_0, W_1)$.

The output of the simulator is computationally indistinguishable from the view of the receiver. Indeed, an adversary that distinguishes between the simulated and the real experiment also distinguishes between when W_{1-b} is uniformly distributed and when $W = h(f, f^{-1}(S(f, z_{1-b}))$, thus contradicting the assumption that h is an hard-core predicate for the enhanced trapdoor OWPs f. □

B.2 Secure 3-Round OT

Using a technique very similar to the one proposed in Section 6, we now propose a protocol $\Pi^{\mathsf{sh-R}}_{\mathcal{OT}}$ that is secure against semi-honest receiver and retains the privacy of receiver against any malicious sender (we refer to the theorems below for the formal statements on the security enjoied by our protocol). We propose only the formal description of the protocol $\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}} := (S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}, R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}})$ in Figure 4 (and the formal security proof) since the techniques are very similar to the techniques used in Section 6. We refer the reader to the technical overview or to Section 6 for an informal discussion of our techniques.

Theorem 12. *If enhanced permutations with efficiently sampleable range and domain exist, then* $\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$ *securely realizes the oblivious transfer functionality* $F_{\mathcal{OT}}$ *against semi-honest receivers and*

$$\{\mathsf{View}^{R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}}_{\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}, S_{\overline{\mathcal{OT}}^{\star}(z)}}(l_0, l_1, 0)\}_{z\in\{0,1\}^{\star}} \approx \{\mathsf{View}^{R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}}_{\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}, S^{\mathsf{sh-R}\star}_{\overline{\mathcal{OT}}}(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^{\star}}$$

Moreover, $\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$ *makes black-box use of the underlying primitive.*

Proof. **Security against malicious sender.** Before proving this part of the theorem, we prove the following lemma.

Lemma 2. *For every non-uniform* PPT *adversary* $S^{\star}_{\mathcal{OT}}$ *controlling the sender, if one of the following holds with overwhelming probability*

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

2. $\text{Prob}\left[\ \mathsf{E_g}^i\ \right] < 2^{-1}$

then

$$\{\mathsf{View}^{R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}}_{\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}, S_{\overline{\mathcal{OT}}^{\star}(z)}}(l_0, l_1, 0)\}_{z\in\{0,1\}^{\star}} \approx \{\mathsf{View}^{R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}}_{\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}, S^{\mathsf{sh-R}\star}_{\overline{\mathcal{OT}}}(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^{\star}}$$

Proof. The first condition implies that the functions sent by the malicious sender in the first round are almost permutations with overwhelming probability. Therefore, we can invoke the Theorem 10 to conclude this part of the proof. If the second condition holds, then with overwhelming probability the collision sets of f has size at least $2^{-1}|D|$. Hence, also in this case we can invoke the Theorem 10 and conclude the proof. □

Common input: Security parameters: $\lambda := 2^\kappa$ for some $k \in \mathbb{N}$, $n := 4\lambda$
Input to $R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$: $b \in \{0,1\}$.
Input to $S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$: $l_0 \in \{0,1\}, l_1 \in \{0,1\}$.

$S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}} \rightarrow R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$
1. Run GL on input $(l_0, l_1, i, \log(n))$ thus obtaining $(l_0^1, l_1^1), \ldots, (l_0^n, l_1^n)$.
2. For $i = 1, \ldots, n$ run $S_{\mathcal{OT}}$ on input (l_0^i, l_1^i) thus obtaining $(\mathsf{aux}_s^i, \mathsf{ot}_1^i)$.
3. Send $\mathsf{ot}_1^1, \ldots, \mathsf{ot}_1^n$ to $R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$.

$R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}} \rightarrow S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$
1. Run GB on input $(b, 1, \log(n))$ thus obtaining $b^1, \ldots, b^n$.
2. For $i = 1, \ldots, n$ run $R_{\mathcal{OT}}$ on input (b^i, ot_1^i) thus obtaining ot_2^i.
3. Send $\mathsf{ot}_2^1, \ldots, \mathsf{ot}_2^n$ to $S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$

$S^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}} \rightarrow R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$
1. For $i = 1, \ldots, n$ run $\Pi_{\mathcal{OT}}$ on input $(\mathsf{ot}_1^i, \mathsf{ot}_2^i, \mathsf{aux}_s^i)$ thus obtaining ot_3^i.
2. Send $\mathsf{ot}_3^1, \ldots, \mathsf{ot}_3^n$ to $R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$.

Output Phase of $R^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$
1. For $i = 1, \ldots, n$ run $R_{\mathcal{OT}}$ on input $(\mathsf{ot}_1^i, \mathsf{ot}_2^i, \mathsf{ot}_3^i)$ and aux_r^i thus obtaining $l^i_{b^i}$.
2. Output $l^1_{b^1} \oplus \cdots \oplus l^n_{b^n}$

$\mathsf{GB}(b, i, n)$
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 $\mathsf{GB}(b_0, i+1, n)$, $\mathsf{GB}(b_1, i+1, n)$.

$\mathsf{GL}((l_0, l_1), i, n)$
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 $\mathsf{GL}((l_{0,0}, l_{0,1}), i+1, n)$, $\mathsf{GL}((l_{1,0}, l_{1,1}), i+1, n)$.

Figure 4: Formal description of $\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$

We are now ready to complete this part of the proof. Let $\Pi^i_{\mathcal{OT}}$ be the i-th execution of $\Pi_{\mathcal{OT}}$ with $i \in \{1, \ldots, 4\lambda\}$ and let f_i be the function chosen by the malicious sender to execute $\Pi^{\mathsf{sh-R}^i}_{\mathcal{OT}}$; we denote with $\mathsf{E_g}^i$ the event in which a uniformly random values r is sampled such that $y \leftarrow S_R(f^i; r)$ is good, with $i \in \{1, \ldots, 4\lambda^2\} \in \{0,1\}$.

As we have proven in Theorem 10, if for some $i \in \{1, \ldots, 4\lambda\}$ we have that

1. f^i is almost a permutations or
2. $f^i \in L$.

then $\Pi^{\mathsf{sh-R}^i}_{\mathcal{OT}}$ is secure against malicious senders and so would be $\Pi^{\mathsf{sh-R}}_{\overline{\mathcal{OT}}}$ because of the security offered by the combiner. Let us now assume that it does not exist i such that at least one of the conditions showed above holds. This means that for all $i = 1, \ldots, 4\lambda$ we have that $|C(f^i)| < 2^{-1}|D|$.

We have that for all $i \in I$ the probability that the values (z_0, z_1) are good is $\Pr[\mathsf{E_{bg}}^i] := \Pr[\mathsf{E_g}^{i,0} \wedge \mathsf{E_g}^{i,1}] \geq 4^{-1}$ (since $\Pr[\mathsf{E_g}^{i,0}] \geq 2^{-1}$ and $\Pr[\mathsf{E_g}^{i,1}] \geq 2^{-1}$).

We are now interested in proving that there exists an index $i \in I$ such that $\Pr[\mathsf{E_{bg}}^{i,\mathsf{d}}]$ is $1-\nu(\lambda)$. This would conclude this part of the proof since, by the Lemma 2, this implies that one of the execution of $\Pi_{\mathcal{OT}}^{\mathsf{sh-R}}$ is secure. The probability that such an execution does not exists is $\Pr[\neg\mathsf{E_{bg}}^{i}]^{4\lambda} = (1-\Pr[\mathsf{E_g}^{i,0} \wedge \mathsf{E_g}^{i,1}])^{4\lambda} < (1-4^{-1})^{4\lambda} < 2^{-\lambda}$.

Security against semi-honest sender. The proof follows similarly to what is described in the proof of Theorem 11. □

C From Bit OT to String OT

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 $l_0 \in \{0,1\}^\kappa, l_0 \in \{0,1\}^\kappa$ with $\kappa \in \mathbb{N}$ (string-OT).

The work of Brassard et al. [BCS96] proposes a way to construct an information theoretically secure string OT protocol from an information theoretically secure bit OT protocol. The idea proposed in [BCS96] 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 [BCS96] to transform our bit-OT protocol $\Pi_{\overline{\mathcal{OT}}}$ into a string-OT protocol $\Pi^\kappa_{\overline{\mathcal{OT}}} := (S^\kappa_{\overline{\mathcal{OT}}}, R^\kappa_{\overline{\mathcal{OT}}})$.

Let $I = \{i_1, \ldots, i_m\}$ be a set such that $1 \leq i_1 < \ldots i_m \leq n$. Let $y \in \{0,1\}^n$, we denote with y^I the concatenation of the bits of y in the positions indexed by I. In [BCS96] the authors proposed instantiations of a function g with the property that for any two subset $I, J \subseteq \{1, \ldots, n\}$ seeing the bits of x_0^I and x_1^J releases information on at most one of $g(x_0)$ or $g(x_1)$. This function $g : \{0,1\}^n \to \{0,1\}^\kappa$ can be instantiated in an information theoretically secure way as showed in [BCS96]. Let $(l_0, l_1) \in \{0,1\}^\kappa \times \{0,1\}^\kappa$ be the input of $S^\kappa_{\overline{\mathcal{OT}}}$ and b be the input of $R^\kappa_{\overline{\mathcal{OT}}}$. The protocol $\Pi^\kappa_{\overline{\mathcal{OT}}}$ works as follows.

1. $S^\kappa_{\overline{\mathcal{OT}}}$ picks $x_0 \xleftarrow{\$} \{0,1\}^n$, $x_1 \xleftarrow{\$} \{0,1\}^n$ such that $g(x_0) = l_0$ and $g(x_1) = l_1$.
2. $S^\kappa_{\overline{\mathcal{OT}}}$ and $R^\kappa_{\overline{\mathcal{OT}}}$ preform n executions of $\Pi_{\overline{\mathcal{OT}}}$ where $S^\kappa_{\overline{\mathcal{OT}}}$ uses (x_0^i, x_1^i) (where x_d^i corresponds to the i-th bit of x_d for all $d = 0, 1$) as input and $R^\kappa_{\overline{\mathcal{OT}}}$ uses b as input.
3. $R^\kappa_{\overline{\mathcal{OT}}}$ upon receiving the output z_i from the i-th execution of $\Pi_{\overline{\mathcal{OT}}}$ for all $i \in \{1, \ldots, n\}$ computes and outputs $g(z_1 || \ldots || z_n)$.

Theorem 13. *If enhanced trapdoor permutations with efficiently sampleable range and domain exist, then $\Pi^\kappa_{\overline{\mathcal{OT}}}$ securely realizes the oblivious transfer functionality $F_{\mathcal{OT}}$ with one-sided simulation with black-box use of the underlying primitive and the input domain of the receiver is $\{0,1\}^\kappa \times \{0,1\}^\kappa$ with $\kappa \in \mathbb{N}$.*

Proof. The proof against malicious sender follows via standard hybrid arguments. That is, we consider the hybrid $\mathcal{H}_i$ where the first i executions of $\Pi_{\overline{\mathcal{OT}}}$ are run using the bit 0 as input and the remaining $n-i$ executions using 1. $\mathcal{H}_0$ represents the real world execution where $R^\kappa_{\overline{\mathcal{OT}}}$ uses the bit 1 and $\mathcal{H}_n$ to the real world execution where $R^\kappa_{\overline{\mathcal{OT}}}$ uses the bit 0. If by contradiction there exists a malicious sender ${R^\kappa_{\overline{\mathcal{OT}}}}^\star$ that distinguishes $\mathcal{H}_0$ from $\mathcal{H}_n$ then there exists an index j such that $\mathcal{H}_j$ and $\mathcal{H}_{j+1}$ are distinguishable. The reduction to the security of $\Pi_{\overline{\mathcal{OT}}}$ would run internally ${S^\kappa_{\overline{\mathcal{OT}}}}^\star$ computing all the $\Pi_{\overline{\mathcal{OT}}}$ executions but the $j+1$-th execution for which he acts as a proxy between ${S^\kappa_{\overline{\mathcal{OT}}}}^\star$ and the challenger of $\Pi_{\overline{\mathcal{OT}}}$.

In the case of a malicious receiver $R^{\kappa}_{\overline{\mathcal{OT}}}{}^{\star}$ we need to simulate its view when it can obtain bits of x_0 and x_1 given access to $F_{\mathcal{OT}}$ from which we can only obtain either l_0 or l_1. Our proof approach follows the one proposed in [GKOV12]. The simulator simply run n instantiations of the simulator Sim of $\Pi_{\overline{\mathcal{OT}}}$ (which exists by definition) following the below strategy. Let T_0, T_1 be two sets indicating the indices for which the malicious receiver queries with input 0 and 1 respectively. Observe that T_0 and T_1 will always be disjoint. For every query made by the malicious receiver we need to provide it with a response. Our strategy would be to provide it with a random bit as long as neither T_0 nor T_1 "biases" g (Definition 2.1 of [BCS96], i.e., no information about either l_0 or l_1 has been revealed by the values of x_0, x_1 provided to the malicious receiver so far). At the first point when T_b (for some $b \in \{0,1\}$) biases g, we query $F_{\mathcal{OT}}$ with b thus obtaining l_b. We then sample x_b consistently with l_b and the values provided to the malicious receiver so far and continue the simulation. By Definition 2.2 of [BCS96], for the encoding function g only one among T_0 and T_1 can bias g. This fact allows us to continue choosing bits in x_{1-b} at random without querying for l_{1-b}. We note that to prove that this simulator is valid we need to show the view of $R^{\kappa}_{\overline{\mathcal{OT}}}{}^{\star}$ in the real world is indistinguishable from the output of the simulator. This part of the proof follow almost the same arguments used in the second part of the proof of Theorem 8 given that $\overline{\mathsf{Sim}}$ maintains the main thread. Note also that our simulator maintains the main thread as well.

□

D Security Proofs

D.1 Proof of Theorem 6

Proof. To prove the security against a malicious receiver $R^{\star}_{\mathcal{OT}}$ we need to first show how the simulator Sim works. Sim extracts the input b of $R^{\star}_{\mathcal{OT}}$ by rewinding $\Pi_{\mathsf{c\&o}}$ and calls the ideal functionality thus obtaining l_b. At this point Sim uses l_b to compute W^1_b according to the description of $\Pi_{\mathcal{OT}}$ and sets W^1_{1-b} to a random bit.

More formally, we now show a PPT simulator Sim that, having only access to the ideal world functionality $F_{\mathcal{OT}}$, can simulate the output of any malicious $R^{\star}_{\mathcal{OT}}$ running one execution of $\Pi_{\mathcal{OT}}$ with an honest sender $S_{\mathcal{OT}}$. The simulator Sim works as follows. Having oracle access to $R^{\star}_{\mathcal{OT}}$, Sim runs as acts as $S_{\mathcal{OT}}$ until the third round thus sending the functions $f_{0,0}, f_{0,1}, f_{1,0}, f_{1,1}$, β and the two random strings R_0 and R_1. Let c, d be the messages sent in the third round by $R^{\star}_{\mathcal{OT}}$. Sim computes $\delta||r_0||r_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(x, w, c)$ and check if $\mathsf{V}_2(\gamma, \beta, \delta, r_0||r_1) = 0$. If it is then Sim outputs $\perp$ and stops, otherwise Sim rewinds $R^{\star}_{\mathcal{OT}}$ by sending two fresh random strings $\overline{R}_0$ and $\overline{R}_1$ such that $\overline{R}_0 \neq R_0$ and $\overline{R}_1 \neq R_1$, and a freshly generated $\overline{\beta} \xleftarrow{\$} \mathsf{V}_0(1^{\lambda})$.

Let $\overline{c}, \overline{\mathsf{d}}$ be the messages sent in the third round by $R^{\star}_{\mathcal{OT}}$ after this rewind. Sim computes $\overline{\delta}||\overline{r}_0||\overline{r}_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(\overline{x}, \overline{w}, \overline{c})$ (note that if $\overline{\mathsf{d}} \neq \mathsf{d}$ then also the statement and the witness for the DWE scheme change during the rewinding thread). If $\mathsf{V}_2(\gamma, \overline{\beta}, \overline{\delta}, \overline{r}_0||\overline{r}_1) = 0$ (or $R^{\star}_{\mathcal{OT}}$ does not reply send the third round), then we use the following standard argument. If p is the probability of $R^{\star}_{\mathcal{OT}}$ of giving an accepting third round[9], λ/p rewinds are made until $R^{\star}_{\mathcal{OT}}$ gives another answer.

Once that a new third round is received and $\mathsf{V}_2(\gamma, \overline{\beta}, \overline{\delta}, \overline{r}_0||\overline{r}_1) = 1$ then there are only two cases that could occur due to the *existence of committing branch* property of $\Pi_{\mathsf{c\&o}}$:

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

[9] We assume that p is non-negligible since the proof for the case where p is negligible is trivial.

That is, $R^\star_{\mathcal{OT}}$ can either *open* to the same messages r_0 and r_1, or change at most one of the opened messages. This yields to the following important observation. If one among r_0 and r_1 changes during the rewind, let us say $r_{b^\star}$ for $b^\star \in \{0,1\}$ (case 1), then the input bit used by $R^\star_{\mathcal{OT}}$ has to be $b^\star$. Intuitively, this comes from the fact that the only efficient way (i.e., without inverting the TDP) for a receiver to get the output is by picking a message $r_{b\star}$ such that he knows how to compute the inverse of one between $S(f_{\mathsf{d},0}, R_0 \oplus r_0)$ and $S(f_{\mathsf{d},1}, R_1 \oplus r_1)$. Therefore, the simulator invokes the ideal world functionality $F_{\mathcal{OT}}$ using $b^\star$ as input, and upon receiving $l_{b^\star}$ computes $W_{b^\star} = l_{b^\star} \oplus h(f^{-1}_{\mathsf{d},b^\star}(S(f_{\mathsf{d},b^\star}, r_{b^\star} \oplus R_{b^\star})))$ and sets $W_{1-b^\star}$ to a random bit. Then Sim goes back to the main thread and sends W_0 and W_1 to $R^\star_{\mathcal{OT}}$ in the last round.

In the case where the opening of $\Pi_{\mathsf{c\&o}}$ stays the same after the rewinding procedure (case two), Sim just picks a random bit $b^\star\{0,1\}$ and acts as described above.

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

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

  Due to the existence of committing branch property of $\Pi_{\mathsf{c\&o}}$ we have that there exits $b^\star \in \{0,1\}$ such that $\overline{r}_{1-b^\star} = r_{1-b^\star}$. After the rewind $\mathcal{H}_1$ goes back to the main thread and computes $W_{1-b^\star} \xleftarrow{\$} \{0,1\}^\lambda$ and $W_{b^\star} = l_{b^\star} \oplus h(f^{-\lambda}_{\mathsf{d},b^\star}(S(f_{\mathsf{d},b^\star}, r_{b^\star} \oplus R_{b^\star})))$ and sends (W_0, W_1) to $R^\star_{\mathcal{OT}}$. The difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is that in the latter hybrid experiment $W_{1-b^\star}$ is a random bit whereas in $\mathcal{H}_1$ $W_{1-b^\star} = l_{1-b^\star} \oplus h(f^{-\lambda}_{\mathsf{d},1-b^\star}(S(f_{\mathsf{d},1-b^\star}, r_{1-b^\star} \oplus R_{1-b^\star})))$.

  We now prove that the indistinguishability between $\mathcal{H}^{\mathsf{rew}}_0$ and $\mathcal{H}_1$ comes from the security of the hardcore bit function h for the TDP $\mathcal{F}$. More precisely, assuming by contradiction that the $\{\mathsf{OUT}_{\mathcal{H}^{\mathsf{rew}}_0, R^\star_{\mathcal{OT}}(z)}(1^\lambda)\}$ and $\{\mathsf{OUT}_{\mathcal{H}_1, R^\star_{\mathcal{OT}}(z)}(1^\lambda)\}$ are distinguishable and construct an adversary $\mathcal{A}^{\mathcal{F}}$ that distinguishes between the output of $h(x)$ and a random bit having as input r such that $y \xleftarrow{\$} S(f_{\mathsf{d},1-b^\star}; r)$ and $f^{-1}_{\mathsf{d},1-b^\star}(y) = x$.

  $\mathcal{A}^{\mathcal{F}}$, on input the challenge (f, r, u) executes the following steps.

  1. Start $R^\star_{\mathcal{OT}}$ and upon receiving γ pick $(f_{0,0}, f^{-1}_{0,0}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $(f_{0,1}, f^{-1}_{0,1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $(f_{1,0}, f^{-1}_{1,0}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $(f_{1,1}, f^{-1}_{1,1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $\beta \xleftarrow{\$} \mathsf{V}_0(1^\lambda)$, $R_0 \xleftarrow{\$} \{0,1\}^\lambda$, $R_1 \xleftarrow{\$} \{0,1\}^\lambda$

and send $\{f_{i,j}\}_{i,j\in\{0,1\}}, \beta, R_0, R_1$ to $R_{\mathcal{OT}}$.

2. Upon receiving (c, d) from $R^{\star}_{\mathcal{OT}}$ set $x = (f_{1-\mathsf{d},0}, f_{1-\mathsf{d},1})$, $w = (\rho_{1-\mathsf{d},0}, \rho_{1-\mathsf{d},1})$ and compute $\delta||r_0||r_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(x, w, c)$. If $\mathsf{V}_2(\gamma, \beta, \delta, r_0||r_1) = 0$ then output a random bit and stop. Otherwise rewinds $R_{\mathcal{OT}}$ by sending a freshly generated $\{\overline{f}_{i,j}\}_{i,j\in\{0,1\}}, \overline{\beta}, \overline{R}_0, \overline{R}_1$ and repeat this process until $R^{\star}_{\mathcal{OT}}$ answer again the third round with some values $\overline{c}, \overline{\mathsf{d}}$ such that $\mathsf{V}_2(\gamma, \overline{\beta}, \overline{\delta}, \overline{r}_0||\overline{r}_1) = 1$ with $\overline{\delta}||\overline{r}_0||\overline{r}_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(\overline{x}, \overline{w}, \overline{c})$, $\overline{x} = (\overline{f}_{1-\mathsf{d},0}, \overline{f}_{1-\mathsf{d},1})$, $\overline{w} = (\overline{\rho}_{1-\mathsf{d},0}, \overline{\rho}_{1-\mathsf{d},1})$.
3. Check if there exists $t \in \{0,1\}$ such that $\overline{r}_t \neq r_t$. If such a bit exists then set $b^{\star} := t$, otherwise sample $b^{\star} \xleftarrow{\$} \{0,1\}$. Computes the following steps.
   3.1. Pick a random bit $\mathsf{j} \in \{0,1\}$, three random strings $\rho'_{\mathsf{j},b^{\star}}, \rho'_{1-\mathsf{j},0}, \rho'_{1-\mathsf{j},1} \xleftarrow{\$} \{0,1\}^{\lambda}$, set $g_{\mathsf{j},1-b^{\star}} := f$ and compute $(g_{\mathsf{j},b^{\star}}, g^{-1}_{\mathsf{j},b^{\star}}) \xleftarrow{\$} \mathsf{Gen}(1^{\lambda}, \rho'_{\mathsf{j},b^{\star}})$, $(g_{1-\mathsf{j},0}, g^{-1}_{1-\mathsf{j},0}) \xleftarrow{\$} \mathsf{Gen}(1^{\lambda}, \rho'_{1-\mathsf{j},0})$, $(g_{1-\mathsf{j},1}, g^{-1}_{1-\mathsf{j},1}) \xleftarrow{\$} \mathsf{Gen}(1^{\lambda}, \rho'_{1-\mathsf{j},1})$, $\beta \xleftarrow{\$} \mathsf{V}_0(1^{\lambda})$.
   3.2. Compute $R'_{1-b^{\star}} \leftarrow r_{1-b^{\star}} \oplus r$, $R'_{b^{\star}} \xleftarrow{\$} \{0,1\}^{\lambda}$, $\beta' \xleftarrow{\$} \mathsf{V}_0(1^{\lambda})$.
   3.3. Rewind $R^{\star}_{\mathcal{OT}}$ up to the second round and send $\{g_{i,j}\}_{i,j\in\{0,1\}}, \beta', R'_0, R'_1$ to him.
4. Upon receiving (c', d') from $R^{\star}_{\mathcal{OT}}$ if $\mathsf{d} \neq \mathsf{j}$ then output a random bit and stop. Otherwise set $x' := (g_{1-\mathsf{j},0}, g_{1-\mathsf{j},1})$, $w' := (\rho'_{1-\mathsf{j},0}, \rho'_{1-\mathsf{j},1})$ and compute $\delta'||r'_0||r'_1 \leftarrow \mathsf{Dec}^{\mathsf{2f}}(x', w', c')$.
5. If $\mathsf{V}_2(\gamma, \beta', \delta, r'_0||r'_1) = 0$ then repeat the step 3.1, otherwise continue with the following steps.
6. If $r'_{1-b^{\star}} \neq r_{1-b^{\star}}$ then stop and output a random bit, otherwise set $W_{1-b^{\star}} \leftarrow u \oplus l_{1-b^{\star}}$ and compute $\omega_{b^{\star}} \leftarrow g^{-1}_{\mathsf{d},b^{\star}}(S(\overline{r}_{b^{\star}} \oplus \overline{R}_1))$, $W_{b^{\star}} = l_{b^{\star}} \oplus h(f_{\mathsf{d},b^{\star}}, \omega_{b^{\star}})$ and send (W_0, W_1) to $R^{\star}_{\mathcal{OT}}$.
7. Output what $R^{\star}_{\mathcal{OT}}$ outputs.

Note that if $u = h(x)$ then $R^{\star}_{\mathcal{OT}}$ acts as in $\mathcal{H}^{\mathsf{rew}}_0$, otherwise $R^{\star}_{\mathcal{OT}}$ acts as in $\mathcal{H}_1$. By contradiction, we are assuming that there exists an adversary $\mathcal{A}$ that has a non-negligible advantage adv in distinguishing H^{rew}_0 from H_1. Since the probability that the $\mathcal{A}^{\mathcal{F}}$ guesses the correct $b^{\star}$ and a bit j such that $\mathsf{j} = \mathsf{d}$ is $1/4$, then we can claim that $\mathcal{A}^{\mathcal{F}}$ breaks the security of the hardcore bit predicate with probability $1/2 + \mathsf{adv}/4$, thus reaching a contradiction.

□

D.2 Proof of Theorem 7

Proof. Let d be the bit sent by $R_{\mathcal{OT}}$ in the third round. We divide the proof is two parts, one for each of the case listed above.

Case 1. Note that in this case we have the guarantee that the functions $f_{\mathsf{d},0}$ and $f_{\mathsf{d},1}$ are almost permutations, therefore the proof can proceed as follows.

We consider the experiment H_0 where $R_{\mathcal{OT}}$'s input is 0 and the experiment H_1 where $R_{\mathcal{OT}}$'s input is 1 and we prove that $S^{\star}_{\mathcal{OT}}$ cannot distinguish between H_0 and H_1. More precisely, in the experiment H_1 the bit 1 instead of the bit 0 is used to compute the messages of $\Pi_{\mathsf{c\&o}}$. To prove that $H_0 \approx H_1$ we can rely on the fact that $f_{\mathsf{d},0}$ and $f_{\mathsf{d},1}$ are almost permutations with efficiently samplable domain and range and on the committing branch indistinguishability property of $\Pi_{\mathsf{c\&o}}$.

Case 2. In this case we can rely on the hiding of $(\mathsf{Enc}^{\mathsf{2f}}, \mathsf{Dec}^{\mathsf{2f}})$ since $(f_{1-\mathsf{d},0}, f_{1-\mathsf{d},1}) \in L^{\mathsf{2f}}_0$. Therefore, the encryption sent in the third round hides in an information theoretical sense all the

values computed by $R_{\mathcal{OT}}$ that depends on the functions $f_{\mathsf{d},0}$ and $f_{\mathsf{d},1}$ (which might not be injective and thus leaking the bit of the receiver).

The prof proceeds similarly to the proof for the case 1 with the following differences. We consider the hybrid $H_0^{\perp}$ that is exactly like H_0 except for the fact that c contains and encryption of 0^{λ}. $H_0 \equiv H_0^{\perp}$ due to the hiding of the DWE scheme. Similarly, we consider the hybrid $H_1^{\perp}$ that is exactly like H_1 except for the fact that c contains an encryption of 0^{λ}. To prove that $H_0^{\perp} \approx H_1^{\perp}$ we just rely on the committing branch indistinguishability property of $\Pi_{\mathsf{c\&o}}$, and this would conclude the proof.

□

D.2.1 Proof of Theorem 8

Proof. **Security against malicious sender.** Before proving this part of the theorem, we prove the following lemma.

Proof. The first condition implies that the functions sent by the malicious sender in the first round are almost permutations with overwhelming probability. Therefore, we can invoke the Theorem 7 to conclude this part of the proof. If the second condition holds, then with overwhelming probability the collision sets of $f_{0,j}$ and $f_{1,i}$, for some $i, j \in \{0,1\}$, have size at least $2^{-1}|D|$. Hence, also in this case we can invoke the Theorem 7 and conclude the proof. □

We are now ready to complete this part of the proof. Let $\Pi^i_{\mathcal{OT}}$ be the i-th execution of $\Pi_{\mathcal{OT}}$ with $i \in \{1, \ldots, 4\lambda^2\}$ and let $(f^i_{0,0}, f^i_{0,1}, f^i_{1,0}, f^i_{1,1})$ be the four functions chosen by the malicious sender to execute $\Pi^i_{\mathcal{OT}}$; we denote with $\mathsf{E_g}^{i,d,b}$ the event in which a uniformly random values r is sampled such that $y \leftarrow S_R(f^i_{d,b}; r)$ is good, with $i \in \{1, \ldots, 4\lambda^2\}, d, b \in \{0,1\}$.

As we have proven in Sec. 5, if for some $i \in \{1, \ldots, 4\lambda^2\}$ we have that

1. $f^i_{0,0}$ and $f^i_{0,1}$ and $f^i_{1,0}$ and $f^i_{1,1}$ are almost permutations or
2. $(f^i_{0,0}, f^i_{0,1}) \in L_0^{\mathsf{2f}}$ and $(f^i_{1,0}, f^i_{1,1}) \in L_0^{\mathsf{2f}}$.

then $\Pi^i_{\mathcal{OT}}$ is secure against malicious senders and so would be $\Pi_{\overline{\mathcal{OT}}}$ because of the security offered by the combiner. Let us now assume that it does not exist i such that at least one of the conditions showed above holds. This means that for all $i = 1, \ldots, 4\lambda^2$ there exists a bit $\mathsf{d}'_i \in \{0,1\}$ such that $|C(f^i_{1-\mathsf{d}'_i,0})| < 2^{-1}|D|$ and $|C(f^i_{1-\mathsf{d}'_i,1})| < 2^{-1}|D|$.

Let d_i be the selection bit used by the receiver in the i-th execution of $\Pi_{\mathcal{OT}}$; we denote with Econds_i the event in which in the i-th execution of $\Pi_{\mathcal{OT}}$ we have that $|C(f^i_{\mathsf{d}_i,0})| < 2^{-1}|D|$ and $|C(f^i_{\mathsf{d}_i,1})| < 2^{-1}|D|$ (i.e., $\mathsf{d}_i = 1 - \mathsf{d}'_i$). Since d_i is randomly (and independently) chosen by the receiver in each execution $\Pi_{\mathcal{OT}}$ we have that $\Pr[\mathsf{Econds}_i] = 2^{-1}$.

Hence the probability that there exists at least an index $j_1 \in \{1, \ldots, \lambda\}$ in the first λ executions of $\Pi_{\mathcal{OT}}$ such that Econds_{j_1} holds is $\Pr[\exists j_1 : \mathsf{Econds}_{j_1}] \geq 1 - 2^{-\lambda}$. Generalizing, we have that the probability that there exists an index $j_k \in \{1 + \lambda k, \ldots, \lambda(k+1)\}$ for all $k = 0, \ldots 4\lambda - 1$ such that Econds_{j_k} occurs is $\Pr[\exists j_k : \mathsf{Econds}_{j_k}] = 1 - 2^{-\lambda}$. Let $I := \{i_0, \ldots, i_{4\lambda-1}\}$ the set containing all these indexes.

We have that for all $i \in I$ the probability that the values $S_R(f^i_{\mathsf{d}_i,0}, r^i_0 \oplus R^i_0)$ and $S_R(f^i_{\mathsf{d}_i,1}, r^i_1 \oplus R^i_0)$ are good is $\Pr[\mathsf{E_{bg}}^{i,\mathsf{d}}] := \Pr[\mathsf{E_g}^{i,\mathsf{d},0} \wedge \mathsf{E_g}^{i,\mathsf{d},1}] \geq 4^{-1}$ (since $\Pr[\mathsf{E_g}^{i,\mathsf{d},0}] \geq 2^{-1}$ and $\Pr[\mathsf{E_g}^{i,\mathsf{d},1}] \geq 2^{-1}$).

We are now interested in proving that there exists an index $i \in I$ such that $\Pr[\mathsf{E}_{\mathsf{bg}}{}^{i,\mathsf{d}}]$ is $1-\nu(\lambda)$. This would conclude this part of the proof since, by the Lemma 1, this implies that one of the execution of $\Pi_{\mathcal{OT}}$ is secure. The probability that such an execution does not exists is $\Pr[\neg\mathsf{E}_{\mathsf{bg}}{}^{i}]^{4\lambda} = (1-\Pr[\mathsf{E}_{\mathsf{g}}{}^{i,0} \wedge \mathsf{E}_{\mathsf{g}}{}^{i,1}])^{4\lambda} < (1-4^{-1})^{4\lambda} < 2^{-\lambda}$.

Security against malicious receiver. We want to prove that for every non-uniform PPT adversary $R^{\star}_{\overline{\mathcal{OT}}}$ controlling the receiver in the real model, there exists a non-uniform PPT adversary $\overline{\mathsf{Sim}}$ for the ideal model such that $\{\mathsf{REAL}_{\Pi_{\overline{\mathcal{OT}}},R^{\star}_{\overline{\mathcal{OT}}}(z)}(1^{\lambda})\}_{z\in\{0,1\}^{\lambda}} \approx \{\mathsf{IDEAL}_{F_{\mathcal{OT}},\overline{\mathsf{Sim}}(z)}(1^{\lambda})\}_{z\in\{0,1\}^{\lambda}}$.

As showed in the proof of Theorem 6, for every non-uniform PPT adversary $R^{\star}_{\mathcal{OT}}$ controlling the receiver in the real model, there exists a non-uniform PPT adversary Sim for the ideal model such that

$\{\mathsf{REAL}_{\Pi_{\mathcal{OT}},R^{\star}_{\mathcal{OT}}(z)}(1^{\lambda})\}_{z\in\{0,1\}^{\lambda}} \approx \{\mathsf{IDEAL}_{F_{\mathcal{OT}},\mathsf{Sim}(z)}(1^{\lambda})\}_{z\in\{0,1\}^{\lambda}}$. Moreover, Sim maintains the main thread.

The simulator $\overline{\mathsf{Sim}}$ internally uses Sim for any of the executions of $\Pi_{\mathcal{OT}}$. More formally $\overline{\mathsf{Sim}}$ works as follows.

1. Run the adversary $\mathcal{A}$ and for each message that refers to a new execution of $\Pi^{i}_{\mathcal{OT}}$ start a new instance of Sim and act as a proxy between $\mathcal{A}$ and Sim for all the messages that refer to $\Pi^{i}_{\mathcal{OT}}$.
2. Upon receiving the bits $b^1,\ldots,b^{4\lambda^2}$ from the simulators[10] compute $b = b^1 \oplus \cdots \oplus b^{4\lambda^2}$ and send b to $F_{\mathcal{OT}}$.
3. Upon receiving l_b from $F_{\mathcal{OT}}$ compute a random secret sharing $(l^1_b,\ldots,l^{4\lambda^2}_b)$ such that $l_b = (l^1_b \oplus \cdots \oplus l^{4\lambda^2}_b)$
4. For $i = 1,\ldots,4\lambda^2$ keep running the i-th simulated execution using l^i_b as input.
5. Output the concatenation of the outputs of all the simulators.

The proof proceeds via hybrid arguments. More precisely, we start by consider the hybrid experiment $\mathcal{H}_0$ that corresponds to the real world execution where the input (l_0,l_1) is used by $S_{\overline{\mathcal{OT}}}$ and then we consider the experiment $\mathcal{H}_i$ where the simulator Sim (instead of the honest sender procedure) is used in the first i parallel executions of $\Pi_{\mathcal{OT}}$. Supposing by contradiction that the output distributions of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ (for some $i \in \{1, 4\lambda^2-1\}$) are distinguishable, then we can construct a malicious receiver $R^{\star}_{\overline{\mathcal{OT}}}$ that breaks the security of $\Pi_{\mathcal{OT}}$ against malicious senders. We note that the fact that the first i executions of $\Pi_{\mathcal{OT}}$ are replaced with executions of Sim could disturb the reduction to the security of $\Pi_{\mathcal{OT}}$ when proving that the output distribution of $\mathcal{H}_i$ is indistinguishable from the output distribution of $\mathcal{H}_{i+1}$ (this because of the rewinds made by Sim) and this requires some care. More precisely, as described in the security proof of $\Pi_{\mathcal{OT}}$, the simulation made by Sim roughly works by rewinding from the third to the second round. After that Sim goes back to the main thread and from this moment onwards Sim works straight line. The feature that we use in this proof to avoid issue due to the rewinds made by the execution of the simulators of Sim is the fact that Sim maintains the main thread. Let $\mathcal{C}^{\mathcal{OT}}$ be the challenger of $\Pi_{\mathcal{OT}}$ against malicious receiver; our adversary $R^{\star}_{\mathcal{OT}}$ works as following.

1. Upon receiving the first round of $\Pi_{\overline{\mathcal{OT}}}$ from $R^{\star}_{\overline{\mathcal{OT}}}$, forward the $(i+1)$-th component ot^{i+1}_1 (that refers to the first message of $\Pi^{i+1}_{\mathcal{OT}}$) to $\mathcal{C}^{\mathcal{OT}}$[11].
2. Upon receiving ot^{i+1}_2 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^{\star}_{\overline{\mathcal{OT}}}$ by computing the second round of

[10] Note that the i-th execution of Sim outputs a bit b_i if the adversary is non-aborting as proved in Theorem 6.

[11] We recall that $\Pi_{\overline{\mathcal{OT}}}$ is constructed by executing in parallel $4\lambda^2$ instantiations of $\Pi_{\mathcal{OT}}$, therefore in this reduction we are just replacing the $(i+1)$-th component of every rounds sent to $R^{\star}_{\overline{\mathcal{OT}}}$ with the value received by $\mathcal{C}^{\mathcal{OT}}$. Vice versa, we forward to $\mathcal{C}^{\star}$ the $(i+1)$-th component of the rounds received from $R^{\star}_{\overline{\mathcal{OT}}}$.

$\Pi_{\overline{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_2 is used.

3. Upon receiving the third round of $\Pi_{\overline{\mathcal{OT}}}$ from $R^{\star}_{\overline{\mathcal{OT}}}$, forward the $(i+1)$-th component ot_3^{i+1} to $\mathcal{C}^{\mathcal{OT}}$.
4. Upon receiving ot_4 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^{\star}_{\overline{\mathcal{OT}}}$ by computing the fourth round of $\Pi_{\overline{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_4^{i+1} is used.
5. Output what $R^{\star}_{\overline{\mathcal{OT}}}$ outputs.

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

□

D.3 Proof of Theorem 4

Proof. The completeness follows immediately from the definition of trapdoor permutation. To prove the hiding property we proceed as follows.

We say that a value $y \in D_\alpha$ is *bad* if there exist at least one pair $(x^0, x^1) \in D_\alpha \times D_\alpha$ such that $y = f_\alpha(x^0) = f_\alpha(x^1)$ and $x^0 \neq x^1$. We denote with $\mathsf{E_g}$ the event in which a uniformly random values x is sampled (i.e., $x \xleftarrow{\$} S(\alpha)$) and $f_\alpha(x)$ is good. We denote with $\mathsf{GoodGuess}_i$ the event where the values y_i and j_i chosen during the encryption process are such that:

- y_i is bad, hence there exist at least two values x_i and $\tilde{x}_i$ such that $x_i \neq \tilde{x}_i$ and $f_\alpha(x_i) = f_\alpha(\tilde{x}_i)$.
- the value $j_i \in \{1, \ldots, \lambda\}$ chosen during the encryption is such that if x_i is parsed as $x_i^1, \ldots x_i^{j_i}, \ldots x_i^\lambda$ and $\tilde{x}_i$ is parsed as $\tilde{x}_i^1, \ldots \tilde{x}_i^{j_i}, \ldots \tilde{x}_i^\lambda$ then $x_i^{j_i} \neq \tilde{x}_i^{j_i}$.

We can compute $\Pr[\mathsf{GoodGuess}_i] = \Pr[\mathsf{E_b}]\lambda^{-1} \geq 2^{-1}\lambda^{-1}$, where λ^{-1} represents the probability of guessing the index j_i where the two values x_i and $\tilde{x}_i$ differ.

To conclude the proof we just need to show that $\Pr[\exists k \text{ s.t. } \mathsf{GoodGuess}_k] \geq 1 - \nu(\lambda)$. We note that this would be sufficient to conclude the proof since in this case ct represents a valid encryption of both 0 and 1. Indeed, we have that $\Pr[\exists k \text{ s.t. } \mathsf{GoodGuess}_k] = 1 - \Pr[\forall k \in \{1, \ldots, n\} \neg \mathsf{GoodGuess}_k] = 1 - \prod_{i=1}^{n} \Pr[\neg \mathsf{GoodGuess}_i] \geq 1 - \left(1 - \frac{1}{2\lambda}\right)^n \geq 1 - e^{-\lambda}$

□