J Cryptol (2018) 31:671–697
https://doi.org/10.1007/s00145-017-9267-7

Journal of
CRYPTOLOGY

Completeness for Symmetric Two-Party Functionalities: Revisited*

Yehuda Lindell
Department of Computer Science, Bar Ilan University, Ramat Gan, Israel
lindell@biu.ac.il

Eran Omri†
Department of Computer Science, Ariel University, Ariel, Israel
omrier@ariel.ac.il

Hila Zarosim
Department of Computer Science, Bar Ilan University, Ramat Gan, Israel
hila.zarosim@gmail.com

Communicated by Omer Reingold.

Received 10 April 2014 / Revised 2 March 2016
Online publication 2 November 2017

Abstract. Understanding the minimal assumptions required for carrying out cryptographic tasks is one of the fundamental goals of theoretic cryptography. A rich body of work has been dedicated to understanding the complexity of cryptographic tasks in the context of (semi-honest) secure two-party computation. Much of this work has focused on the characterization of trivial and complete functionalities (resp., functionalities that can be securely implemented unconditionally, and functionalities that can be used to securely compute all functionalities). Most previous works define reductions via an ideal implementation of the functionality; i.e., f reduces to g if one can implement f using a black-box (or oracle) that computes the function g and returns the output to both parties. Such a reduction models the computation of f as an *atomic operation*. However, in the real world, protocols proceed in rounds, and the output is not learned by the parties simultaneously. In this paper, we show that this distinction is significant. Specifically, we show that there exist symmetric functionalities (where both parties receive the same outcome) that are neither trivial nor complete under "black-box reductions," and yet the existence of a constant-round protocol for securely computing such a functionality implies infinitely often oblivious transfer (meaning that it is secure for infinitely many values of the security parameter). In light of the above, we propose an alternative definitional infrastructure for studying the triviality and completeness of functionalities.

Keywords. Secure computation, Hardness assumptions, Completeness, Non-black-box reductions.

*This work was supported by the ISRAEL SCIENCE FOUNDATION (Grant No. 189/11). Hila Zarosim is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. Most of this work was done while Eran Omri was at Bar-Ilan University. A preliminary version appeared in Asiacrypt 2012 [19].

† E. Omri: Partially supported by ISF Grant 544/13.

1. Introduction

1.1. *Secure Computation and Completeness*

In the setting of secure two-party computation, two parties with respective private inputs x and y wish to compute a function f of their inputs. The computation should preserve a number of security properties, such as *privacy* (meaning that nothing but the specified output is learned), *correctness*.

In the late 1980s, it was shown that every function can be securely computed in the presence of semi-honest and malicious adversaries, assuming the existence of enhanced trapdoor permutations [7,24]. Soon after, it was shown that any function can be securely computed, *given a black-box for computing the oblivious transfer function* [11]. This work demonstrated that there exist "complete" functions for secure computation, that is, functions that can be used to securely compute all other functions. Such functions are of great interest. On the one hand, when attempting to base secure computation on weaker hardness assumptions, it suffices to construct a secure protocol for a complete function based on some weaker assumption, since it will imply that this assumption suffices for securely computing all functions. On the other hand, it is immediate that a complete function is the "hardest" to compute, at least with respect to the minimum hardness assumption. Due to the above, much research has been carried out in an attempt to classify functions as complete or not, and as trivial or not (where triviality means that it can be securely computed without any assumption).

The Complexity of Secure Computation. Currently, we have a good picture regarding the complexity of secure computation, through the aforementioned research of completeness. For example, we know that in the setting of asymmetric functionalities (where only one of the two parties receives output), every two-party (deterministic) asymmetric function is either complete or trivial [2,13]. Thus, no non-trivial asymmetric function can be securely computed under an assumption weaker than that needed for securely computing oblivious transfer.

However, in the setting of symmetric functionalities, where both parties receive the same output, the picture is more complex [12,17,21]. For example, unlike the asymmetric setting, there exist (deterministic) symmetric functions that are neither complete nor trivial. This raises the following fundamental question:

> *What hardness assumptions are sufficient and necessary for securely computing functions that are neither complete nor trivial?*

The starting point of this work is the above question. We stress that although Kilian [12] separated these functions from all complete functions, hinting that it may be possible to devise secure protocols for such functions relying on assumptions that are strictly weaker than those needed for oblivious transfer, the only known protocols for securely computing non-trivial functions are general protocols that rely on hardness assumptions that can be used to compute *any* function including oblivious transfer.

Black-Box Reductions and Black-Box Separations. As we have mentioned, a large body of work has been dedicated to understanding the complexity of cryptographic tasks in the context of (semi-honest) secure two-party computation (see, e.g., [2,3,8,11–13,21]).

The idea underlying much of this work is that if the possibility to securely compute a functionality f_1 implies the possibility to securely compute a functionality f_2, then f_1 is at least as hard as f_2. It is then said that f_2 reduces to f_1. A functionality f is called complete if all other functionalities reduce to f. The question of how to define the notion of reduction is of great importance to the implication of these results.

Most previous works define a reduction via an ideal implementation of a function (with [2] being an exception); i.e., f_2 reduces to f_1 if a secure protocol for computing f_2 can be constructed given a black-box (trusted party or oracle) that computes f_1 and gives the outputs to both parties simultaneously.[1] The advantage of (black-box) reductions in the above type is that they always provide a constructive way of securely computing one functionality given an implementation of another. However, the disadvantage of black-box reductions is that a *separation* (i.e., a proof that one function does *not* reduce to another) does not necessarily imply that one cannot construct a secure protocol for one function given a secure protocol from the other. This is due to the fact that a construction may be non-black-box.

1.2. *Our Contributions*

In this work, we give substantial evidence that the picture of the *computational hardness* of securely computing two-party functionalities in the presence of semi-honest adversaries is different from that drawn by the characterizations of completeness of [12,17]. Specifically, we show that there exist symmetric functionalities f (i.e., where both parties get the same output) that are *not black-box-complete* (i.e., OT cannot be implemented using an black-box computing f) but may be in some sense as hard to obtain as OT. Specifically, we prove the following:

Theorem 1.1. (informal) *If there exists a constant-round protocol π that securely computes a symmetric non-trivial functionality f over a constant-size domain, in the presence of semi-honest adversaries, then there exists an infinitely often-OT that is secure in the presence of semi-honest uniform adversaries.*[2]

Needless to say, Theorem 1.1 is of interest for non-trivial functionalities f that are *not* complete; as we have mentioned, such functionalities exist. Our main observation in proving this result is that in real-world protocols, a black-box that simultaneously provides outputs to both parties does not exist. Rather, parties learn their outputs gradually, and hence, in any constant-round protocol, there must be a round in which one party learns substantial information before the other party does. Thus, essentially there is no difference between the symmetric setting (where both parties receive output and there

[1] We stress that the issue of simultaneity has nothing to do with fairness since we consider semi-honest adversaries. Rather, the important point is that both parties receive the same information and it is not possible for one party to learn the output of the function while the other does not. If this were not the case, and only one party receives output, then the symmetric setting reduces to the asymmetric setting where all functionalities are either trivial or complete.

[2] Infinitely often-OT is a protocol for computing OT for which correctness and security hold for infinitely many n's (rather than for all sufficiently large n).

are functions that are neither complete nor trivial) and the asymmetric setting (where only one party receives an output and all functions are either trivial or complete).

Theorem 1.1 can be generalized to deal with all functionalities over a constant-size domain that are not trivial in the presence of semi-honest adversaries, even those that are not symmetric. Namely, a semi-honest secure constant-round protocol for any non-trivial, deterministic, two-party functionality f, over a finite domain, even with $f_{\mathsf{A}} \neq f_{\mathsf{B}}$, implies the existence of an infinitely often-OT that is secure in the presence of semi-honest uniform adversaries. We further explain this generalization in Remark 3.6, appearing in Sect. 3.2.

Alternative Formulation of Completeness—Existential Completeness. In light of the above, we propose a definition of completeness that is not black-box. This definition is in the spirit of the notion of reduction used by Beimel et al. [2] in the setting of asymmetric functionalities. We define the notion of an "achievable class" of a given functionality f. Informally speaking, the achievable class of a functionality f contains all functionalities that can be securely computed, assuming that f can be securely computed. We use this notion in the natural way in order to redefine reductions, and trivial and complete functionalities. Our formulation has the disadvantage of being completely non-constructive. However, it has the advantage of providing a more accurate picture regarding the hardness assumptions required for secure computation.

1.3. *Techniques*

Obtaining OT from an Insecure Minor in the Asymmetric Setting. We first recall that an asymmetric functionality that contains an insecure minor implies oblivious transfer when we consider black-box reductions (see [2] for a full characterization of the asymmetric functionalities). Loosely, an *insecure minor* for a function f consists of four inputs x, x', y, y' such that either $f(x, y) = f(x, y')$ and $f(x', y) \neq f(x', y')$ (this is called an X-minor) or $f(x, y) = f(x', y)$ and $f(x, y') \neq f(x', y')$ (this is called a Y-minor). This notion is formally defined in Definition 3.1 in Sect. 3.1.

Now, assume that a function f contains an X-minor, and let x, x', y, y' be the corresponding inputs (see Fig. 1 for an example of such a function) and assume that only party A learns the output of f (remember that f is asymmetric). We now explain why f implies the variant of OT where the receiver holds a bit c, and the sender holds a bit b. If $c = 0$, the receiver learns b and otherwise it learns nothing (the sender learns nothing)[3]. To obtain an OT protocol from a black-box for computing the asymmetric function f, we let the receiver emulate party A and the sender emulate B. The sender uses y if $b = 0$ and y' if $b = 1$. If $c = 0$, the receiver uses input x and by the structure of f it cannot distinguish between the case that $b = 0$ and the case that $b = 1$ because the output of f is the same for both cases ($f(x, y) = f(x, y') = \alpha$). If $c = 1$, then the receiver uses x', in which case its output is different when $b = 0$ and when $b = 1$ ($f(x', y) \neq f(x', y')$). Note that since f is asymmetric and party B does not learn the output of f, the sender learns nothing about the input of the receiver.

[3] This OT variant, called naïve-OT, is equivalent in the semi-honest setting to the well-known Rabin-OT [23] and 1-out-of-2 OT [4]. Specifically, these are all complete for general two-party computation.

	y	y'
x	α	α
x'	β	γ

Fig. 1. An asymmetric function f with an X-minor ($\beta \neq \gamma$).

	y_1	y_2	y_3
x_1	0	0	1
x_2	3	4	1
x_3	3	2	2

Fig. 2. Kushilevitz's function $f_{\mathcal{KUSH}}$.

Moving to the Symmetric Setting. Note that the above idea does not work for the symmetric variant of f, where both parties learn the output. This is due to the fact that if party B also learns the output of f, then the sender playing the role of B can learn the input of the receiver, contradicting the privacy of the OT protocol.

However, we use the observation that real-life protocols do not behave as black-boxes, in the sense that they are not atomic and the parties do not learn the output simultaneously. Real-life protocols are actually iterative and at any round only one of the parties gets some new information. We use this observation to show that for constant-round protocols for non-trivial symmetric functionalities, there exists a round where one of the parties has substantially more information than the other party. We show that if we stop the execution of the protocol at this round, then we are able to obtain a weak variant of OT, where correctness holds with probability noticeably larger than $1/2$. We then show how to amplify the correctness of this protocol to obtain a fully correct OT protocol.

We demonstrate this idea by considering a *3-round* protocol for a symmetric functionality $f_{\mathcal{KUSH}}$, described in Fig. 2. This is an example of a symmetric function that is neither trivial nor complete (with respect to black-box reductions). Assume that there exists a 3-round protocol π for securely computing $f_{\mathcal{KUSH}}$ and assume that party A is the one sending the first message m_1 in π. Now, it is easy to see that after the first message is sent, B has no information about the input of A. The reason for this is as follows. First, note that A's first message is independent of B's input. Next, if B's input is y_1, then since $f_{\mathcal{KUSH}}(x_2, y_1) = f_{\mathcal{KUSH}}(x_3, y_1)$, the first message m_1 sent by A should be indistinguishable for when the input of A is x_2 and when it is x_3. If B's input is y_3, then since $f_{\mathcal{KUSH}}(x_1, y_3) = f_{\mathcal{KUSH}}(x_2, y_3)$, the first message m_1 should be indistinguishable for when the input of A is x_1 and when it is x_2. Since m_1 is independent of B's input, this implies that m_1 should be indistinguishable for all possible inputs of A and hence it reveals no information about the input of A. On the other hand, after receiving m_2 from B, party A must know the output of the functionality because this is the last message that A receives in π (recall that π is a 3-round protocol where A sends the first message, B the second message and A the third message). Hence, if we stop the execution of π after the second message, we are in the following situation:

- B knows nothing about the input of A,
- A knows the output of the functionality.

These properties in some sense allow us to reduce our problem to the asymmetric case in which we already know how to construct a secure OT protocol provided that the function contains an X-minor. For this matter, we can take x_1, x_2, y_1, y_2 that form an X-minor.

Now, if π is such that party B is the one sending the first message in π, then a similar argument yields that after the second round, A knows nothing about the input of B and B knows the output of the functionality. Here we will need to use a Y-minor in order to obtain a secure OT protocol and having the sender plays the role of A and the receiver plays the role of B. For the Y-minor, we can take x_2, x_3, y_1, y_2.

The above idea works for the case the number of rounds is only 3 and for the specific functionality $f_{\mathcal{KUSH}}$. For the more general case, where we consider any *non-trivial* function and every *constant-round* protocol, we show that we can look at the *first round* in which one of the parties has a noticeable advantage in distinguishing between two possible inputs of the other party. Note that since this is the first round that this happens, the other party has not learned any new information yet and hence we can use the ideas of the asymmetric setting. We show that no matter who this party is, there exists a corresponding minor that can be used in order to construct an OT protocol with weak correctness (the weak correctness is due to the fact that the distinguishing probability is only noticeable). We then use simple repetition to amplify the correctness. Finally, the notion of distinguishing only guarantees a noticeable gap for infinitely many $n \in \mathbb{N}$. Hence, our OT will be an infinitely often-OT.

Limitations of the Techniques. Theorem 1.1 states that it is possible to construct an infinitely often-OT protocol from any constant-round protocol π that securely computes a symmetric non-trivial functionality f over a constant-size domain. This construction has some obvious limitations. First, it requires that the original protocol π has a constant number of rounds, and second, the resulting OT protocol only guarantees that correctness and security hold for infinitely many n's (rather than for all sufficiently large n) against uniform adversaries (see Definition 2.3).

These limitations seem inherent to our techniques. This is further discussed at the end of Sect. 3.3.1. Roughly speaking, these limitations are the result of trying to identify the *first round* in which one of the parties has a noticeable advantage in distinguishing between any two inputs of the other party. If the number of rounds in π is non-constant, i.e., depends on the security parameter n, then this "first" round should become a function of n. Indeed, it is not clear how to define such a function, and moreover, how to prove its existence. Now, in order to argue that a round i is the first round for which a party has a distinguishing advantage, we need to make sure that in any round $j < i$ no party has any distinguishing advantage for all but finitely many n's. Since the converse of that is a distinguishing advantage for infinitely many n's, we define the notion of distinguishing with respect to infinitely many n's. Finally, since the security of the resulting OT protocol is implied by the distinguishing power of the parties of π (which are uniform machines), we only obtain a uniform infinitely often OT protocol.

We remark that, using the results of Kilian [11], one can show that any functionality can be securely computed with uniform infinitely often security (Definition 2.3) given a uniform infinitely often OT protocol. It therefore seems unlikely that such an OT protocol

can be constructed under weaker assumption than fully secure OT (at least, infinitely often secure protocols are not known to be constructible under weaker assumptions, and the known black-box separations for OT [5,9] hold also for infinitely often-OT).

1.4. *Related Work*

As we have already mentioned, completeness in secure two-party computation was investigated in a large body of work [2,3,8,10,12–18,20–22]. We discuss a few that are most relevant to our discussion. Kilian [12] and Kushilevitz [17] consider the symmetric model and give criteria for the existence of unconditionally secure protocols [17] (the same characterization was independently found by Beaver [1]) and for completeness [12]. Maji et al. [21] extended the discussion of the symmetric model to the UC-setting.

The work of Beimel, Malkin, and Micali [2] is most relevant to us. They considered the computational asymmetric model, proving a zero-one law for completeness vs. triviality in this model. Specifically, they define reductions in a non-black-box manner, where f is reducible to g if there exists an efficient transformation that constructs a secure protocol for f from any secure protocol for g. Under this notion of reduction, Beimel et al. [2] show that a function is complete in the asymmetric model if and only if it contains an insecure minor (and that if it is not complete, then it is trivial). Their characterization holds when considering computationally bounded adversaries in either the semi-honest or malicious setting and also when considering unbounded adversaries in the semi-honest setting. Beimel et al. use non-black-box reductions in the asymmetric malicious setting, applying the compiler of [7] that transforms a protocol π for a function f that is secure against semi-honest adversaries, to a protocol π' for f that is secure against malicious adversaries. In this work, we use the notion of non-black-box reductions in the setting of *symmetric* secure two-party computation and consider computationally bounded *semi-honest* adversaries. It is arguably less intuitive that non-black-box reductions should turn out useful in this setting.

Kilian [13] extended the study of completeness for asymmetric deterministic functionalities to the case of computationally unbounded malicious adversaries. The work of [16] gives a general characterization of all finite deterministic 2-party functions that imply a statistically secure implementation of OT, where the resulting OT is secure in the UC framework. This characterization captures also the set of functions that are neither symmetric nor asymmetric (that is, when each party may obtain a different output from the functionality). Almost all of these works consider functions with a constant-size domain and information-theoretic security. The only exception is [8], which deals with security against computationally bounded adversaries for asymmetric functions where the size of the domain may be exponential (in the security parameter).

2. Preliminaries

2.1. *Notations*

A function $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote

probabilistic polynomial time. For an integer ℓ, define $[\ell] = \{1, \ldots, \ell\}$. A *probability ensemble* $X = \{X(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by a and n (the value a will represent the parties' inputs and n the security parameter). All polynomials that we will consider will be with respect to the security parameter, unless explicitly stated; otherwise, specifically, all polynomial time machines will be polynomial in the security parameter. We let λ denote the empty string.

Definition 2.1. Let $X = \{X(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ and $Y = \{Y(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ be two distribution ensembles. The ensembles X and Y are computationally indistinguishable, denoted $X \stackrel{c}{\equiv} Y$, if for every family $\{C_n\}_{n\in\mathbb{N}}$ of polynomial size circuits, there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0, 1\}^*$ and every $n \in \mathbb{N}$:

$$|\Pr[C_n(X(a, n)) = 1] - \Pr[C_n(Y(a, n)) = 1]| < \mu(n).$$

The ensembles X and Y are computationally indistinguishable by uniform machines, denoted $X \stackrel{C}{\equiv}_U Y$, if for every PPT machine D, there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0, 1\}^*$ and every $n \in \mathbb{N}$:

$$\left|\Pr\left[D(X(a, n), 1^n) = 1\right] - \Pr\left[D(Y(a, n), 1^n) = 1\right]\right| < \mu(n).$$

The ensembles X and Y are statistically close, denoted $X \stackrel{s}{\equiv} Y$, if there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0, 1\}^*$ and every $n \in \mathbb{N}$:

$$\mathrm{SD}(\mathrm{X(a, n)}, \mathrm{Y(a, n)}) \stackrel{\mathrm{def}}{=} \frac{1}{2} \cdot \sum_{\alpha} |\Pr[\mathrm{X(a, n)} = \alpha] - \Pr[\mathrm{Y(a, n)} = \alpha]| < \mu(\mathrm{n}).$$

We sometimes consider the above measures for ensembles that are indexed by more restricted subsets of strings (i.e., subsets of $\{0, 1\}^*$) and of natural numbers (i.e., subsets of $\mathbb{N}$). We will say that such ensembles satisfy one of the above closeness measures if they satisfy the appropriate requirement on the restricted set of indices.

2.2. *Secure Two-Party Computation and Oblivious Transfer*

We follow the standard definition of secure two-party computation for semi-honest adversaries, as it appears in [6]. In brief, a two-party protocol π is defined by two interactive probabilistic polynomial time Turing machines A and B. The two Turing machines, called parties, have the security parameter 1^n as their joint input and have private inputs, denoted x and y for A and B, respectively. The computation proceeds in rounds. In each round of the protocol, one of the parties is active and the other party is idle. If party $P \in \{\mathsf{A}, \mathsf{B}\}$ is active in round i, then in this round P writes some value out_P^i on its output tape and sends a message m_i to the other party. Without loss of generality, we assume that A is always active in the odd rounds in π and B in the even rounds. The number of rounds in the protocol is expressed as some function $r(n)$ in the security parameter (where $r(n)$ is bounded by a polynomial).

The view of a party in an execution of the protocol consists of its private input, its random string, and the messages it received throughout this execution. The random variable $\text{VIEW}^{\pi}_{\mathsf{A}}(x, y, 1^n)$ (respectively, $\text{VIEW}^{\pi}_{\mathsf{B}}(x, y, 1^n)$) describes the view of A (resp. B) when executing π on inputs (x, y) (with security parameter n). The output of an execution of π on (x, y) with security parameter n is the pair of values written on the output tapes of the parties when the protocol terminates. This pair is described by the random variable $\text{Output}^{\pi}(x, y, 1^n) = \left(\text{Output}^{\pi}_{\mathsf{A}}(x, y, 1^n), \text{Output}^{\pi}_{\mathsf{B}}(x, y, 1^n)\right)$, where $\text{Output}^{\pi}_{P}(x, y, 1^n)$ is the output of party $P \in \{A, B\}$ in this execution and is implicit in the view of P.

In this work, we consider deterministic functionalities over a finite domain. We therefore provide the definition of security only for deterministic functionalities; see [6] for a motivating discussion regarding the definition.

Definition 2.2. (*security for deterministic functionalities*) A protocol $\pi = (\mathsf{A}, \mathsf{B})$ securely computes a deterministic functionality $f = (f_{\mathsf{A}}, f_{\mathsf{B}})$ in the presence of semi-honest adversaries if the following hold:

Correctness: There exists a negligible function $\mu(\cdot)$, such that for every n and every pair of inputs x, y, it holds that

$$\Pr\left[\text{Output}^{\pi}(x, y, 1^n) = f(x, y)\right] \geq 1 - \mu(n), \tag{1}$$

where the probability is taken over the random coins of the parties.

Privacy: There exist two probabilistic polynomial time (in the security parameter) algorithms $\mathcal{S}_{\mathsf{A}}, \mathcal{S}_{\mathsf{B}}$ (called "simulators"), such that:

$$\left\{\mathcal{S}_{\mathsf{A}}\left(x, f_{\mathsf{A}}(x, y), 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}} \overset{C}{\equiv} \left\{\text{VIEW}^{\pi}_{\mathsf{A}}\left(x, y, 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}}, \tag{2}$$

$$\left\{\mathcal{S}_{\mathsf{B}}\left(y, f_{\mathsf{B}}(x, y), 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}} \overset{C}{\equiv} \left\{\text{VIEW}^{\pi}_{\mathsf{B}}\left(x, y, 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}}. \tag{3}$$

For most of this paper, we will consider functionalities where both parties receive the same output, meaning that $f_{\mathsf{A}} = f_{\mathsf{B}}$. We call such functions symmetric, and we denote by $f(x, y)$ the output that both parties receive. We will also only consider the semi-honest model here and therefore omit this qualification.

*Oblivious Transfer—naïve-*OT *Variant*. The oblivious transfer functionality (OT) is one of the most important cryptographic primitives and is known to be complete for general two-party computation [7,25]. There are several equivalent versions of OT; the most common being Rabin-OT [23] and 1-out-of-2 OT [4]. In this paper, we use a slightly different version presented in [8], called naïve-OT, defined by the functionality $\text{OT}(b, c) = \begin{cases} (\lambda, \lambda) \text{ if } c = 0 \\ (\lambda, b) \text{ if } c = 1 \end{cases}$, meaning that the sender never learns anything (recall that λ is the empty string), and the receiver learns the sender's bit b if its choice-bit c equals 1, but does not learn anything if $c = 0$. This is the same as Rabin-OT except that the receiver chooses whether or not to receive the sender's bit b. In the semi-honest model, it is equivalent to Rabin-OT and to 1-out-of-2-OT.

2.3. *Uniform Infinitely Often Security*

Our main result is a proof that the existence of a constant-round protocol for functionalities that are neither complete nor trivial *almost* implies oblivious transfer. The "almost" in this sentence is due to the fact that we can only prove that it implies oblivious transfer that is secure for *infinitely many n's*, in contrast to all sufficiently large n's. In addition, we can only prove that the oblivious transfer is secure in the presence of uniform distinguishers. This level of security holds if there exists an infinite subset $\mathcal{N} \subseteq \mathbb{N}$ such that Eqs. (1), (2) and (3) from Definition 2.2 hold for *every* $n \in \mathcal{N}$, and Eqs. (2) and (3) hold with respect to uniform distinguishers. We now formally define this weaker notion of security.

Definition 2.3. (*uniform infinitely often security*) A protocol π securely computes a deterministic functionality f in the presence of semi-honest adversaries with uniform infinitely often security if there exists an infinite subset $\mathcal{N} \subseteq \mathbb{N}$ such that

Correctness: There exists a negligible function $\mu(\cdot)$, such that for every $n \in \mathcal{N}$ and every pair of inputs x, y, it holds that

$$\Pr\left[\text{Output}^{\pi}(x, y, 1^n) = f(x, y)\right] \geq 1 - \mu(n),$$

where the probability is taken over the random coins of the parties.

Privacy: There exist two probabilistic polynomial time (in the security parameter) algorithms $\mathcal{S}_{\mathsf{A}}$, $\mathcal{S}_{\mathsf{B}}$ (called "simulators"), such that:

$$\left\{\mathcal{S}_{\mathsf{A}}\left(x, f_{\mathsf{A}}(x, y), 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathcal{N}} \overset{C}{\equiv}_U \left\{\text{VIEW}^{\pi}_{\mathsf{A}}\left(x, y, 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathcal{N}},$$

$$\left\{\mathcal{S}_{\mathsf{B}}\left(y, f_{\mathsf{B}}(x, y), 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathcal{N}} \overset{C}{\equiv}_U \left\{\text{VIEW}^{\pi}_{\mathsf{B}}\left(x, y, 1^n\right)\right\}_{x,y\in\{0,1\}^*;n\in\mathcal{N}}.$$

We stress that there must exist a single $\mathcal{N}$ such that the correctness and privacy conditions all hold for every $n \in \mathcal{N}$ (it does not suffice to require infinitely many n's for which each requirement holds since it is possible that they may hold for different n's in which case the function will be trivial).

3. Our Main Technical Result

In this section, we prove Theorem 1.1. In order to formally state the theorem and our result, we first need to define the class of functions that we consider. We therefore begin with some preliminaries.

3.1. *Preliminaries*

An important property of functionalities that is helpful in our proofs is the existence of an insecure minor, defined below.

	y	y'
x	α	α
x'	β	γ

	y	y'
x	α	β
x'	α	γ

Fig. 3. X-minor and Y-minor (for $\beta \neq \gamma$).

Definition 3.1. (*insecure minor*) Let $f : X \times Y \to R$ be some function. An insecure minor is a tuple of inputs x, x', y, y' such that either

- $f(x, y) = f(x, y')$ and $f(x', y) \neq f(x', y')$ (this is called an X-minor), or
- $f(x, y) = f(x', y)$ and $f(x, y') \neq f(x', y')$ (this is called a Y-minor) (Fig. 3).

Our theorem applies to all non-trivial functionalities, as characterized by Kushilevitz [17]. This characterization uses the notion of "decomposition" of a function. We now define this notion.

Definition 3.2. (*equivalence relation* $\equiv$ *over inputs*) Let $X, Y, Z \subseteq \{0,1\}^*$, and let $f : X \times Y \to Z$. Two inputs $x_1, x_2 \in X$ existentially coincide, denoted $x_1 \sim x_2$, if there exists an input $y \in Y$ such that $f(x_1, y) = f(x_2, y)$. We define an equivalence relation $\equiv$ over X to be the transitive closure of the relation $\sim$ over all $x \in X$. The relations $\sim$ and $\equiv$ are defined over Y similarly.

For example, a triple of inputs x_1, x_2, x_3 is equivalent if there exists y_1 such that $f(x_1, y_1) = f(x_2, y_2)$ and there exists y_2 such that $f(x_2, y_2) = f(x_3, y_2)$. Thus, in the function $f_{\mathcal{KUSH}}$ (see Fig. 2) all $x \in X$ and all $y \in Y$ are equivalent. To see this, note that $f_{\mathcal{KUSH}}(x_1, y_3) = f_{\mathcal{KUSH}}(x_2, y_3)$ and $f_{\mathcal{KUSH}}(x_2, y_1) = f(x_3, y_1)$ and hence x_1, x_2, x_3 are equivalent. Moreover, $f_{\mathcal{KUSH}}(x_1, y_1) = f_{\mathcal{KUSH}}(x_1, y_2)$ and $f_{\mathcal{KUSH}}(x_3, y_2) = f_{\mathcal{KUSH}}(x_3, y_3)$ and hence y_1, y_2, y_3 are equivalent.

Definition 3.3. (*strongly non-decomposable functions*) A function $g : X \times Y \to Z$ is strongly non-decomposable if all $x \in X$ are equivalent, all $y \in Y$ are equivalent, and g is non-constant (i.e., there exist $x, x' \in X$ and $y, y' \in Y$ such that $g(x, y) \neq g(x', y')$).

The binary OR and AND functions are strongly non-decomposable, as is the function $f_{\mathcal{KUSH}}$ defined in Fig. 2 in Sect. 1.3. A strongly non-decomposable function has the property that all inputs are equivalent. We now define a non-decomposable function simply to be a function which has a *subfunction* that is strongly non-decomposable.

Definition 3.4. (*non-decomposable functions*) A symmetric function $f : X \times Y \to Z$ is non-decomposable if there exist $X' \subseteq X$ and $Y' \subseteq Y$ such that f restricted to X' and Y' is strongly non-decomposable; else it is decomposable.

For a better understanding of the origin of the term decomposable, the reader is referred to [17], where it was shown that if a matrix is decomposable, then the function can be computed by an unconditionally secure protocol. Indeed, Kushilevitz [17] proved that a function is trivial if and only if it is decomposable. The function $f_{\mathcal{KUSH}}$ is of particular interest since it is neither trivial (as shown by [17]) nor complete (as shown by [12]).

3.2. *Main Theorem: A Constant-Round Protocol for a Non-Trivial Functionality Implies Infinitely Often OT*

Let f be a symmetric non-decomposable functionality with a finite domain. We show that the existence of a constant-round protocol for computing f implies the existence of a weak variant of oblivious transfer. The idea behind the proof is to run a protocol π for f until the *first* round in which one of the parties learns meaningful information about the input of the other party. Since this is the first round that something is learned and only one party can learn information in any single round, we have that one party has learned something and the other has not. This asymmetry of information suffices for us to construct oblivious transfer.

Our proof proceeds in three stages. First, we prove that a round as described above exists. Intuitively, this is the case since before the protocol execution neither party has any information about the other party's input, but at the end of the execution each party learns significant information about the other party's input. Next, we show that a weak form of oblivious transfer can be constructed from any protocol with such a round (in actuality, we need to prove that such a round exists on a subset of inputs that form an insecure minor, and we demonstrate this in the first step). The OT that we construct is weak in the sense that it is only correct with noticeable probability. Finally, we show how to boost the weak correctness of the OT to fully correct oblivious transfer.

We stress that we do not actually obtain a full oblivious transfer protocol. Rather, our protocol is only secure *infinitely often*; see Definition 2.3. We explain why this is the case at the end of Sect. 3.3.1.

Theorem 3.5. (Theorem 1.1—restated formally) *If there exists a constant-round protocol π that securely computes a symmetric, deterministic, non-decomposable functionality f (over a finite domain) in the presence of semi-honest adversaries, then there exists a uniform infinitely often OT protocol.*[4]

Remark 3.6. (Dealing with general non-trivial functionalities) Theorem 3.5 is stated for symmetric functionalities; however, it can be generalized to all non-trivial functionalities. Indeed, let f be a deterministic, two-party functionality, over a finite domain, that is not necessarily symmetric (i.e., it may be that $f_{\mathsf{A}} \neq f_{\mathsf{B}}$). If f is black-box (semi-honest) complete, then it implies OT by definition.[5] Otherwise, if f is not black-box-complete, then, by the "symmetrization lemma" given in [16, Lemma 1], f is isomorphic to a symmetric functionality f'. This means that f and f' are equivalent up to an invertible consistent renaming of inputs and of (input,output) pairs (see [16] for the formal definition of a consistent renaming). In particular, this means that there exists a constant-round protocol for f if and only if there exists a constant-round protocol for f'. In addition, f is non-trivial if and only if f' is non-trivial.

[4] Since we consider semi-honest security of functions over finite domains, it suffices that the protocol π is secure against *uniform* adversaries for the resulting OT protocol to be infinitely often secure against uniform adversaries.

[5] A characterization of complete deterministic two-party functionalities that are black-box-complete in the semi-honest model was given in [16].

Thus, by Theorem 3.5, it follows that if f is non-trivial in the presence of semi-honest adversaries and there exists a constant-round protocol π that securely computes f in the presence of semi-honest adversaries, then there exists a uniform infinitely often OT protocol.

3.3. *The Proof of Theorem 3.5*

We next present the proof of Theorem 3.5. Recall that a non-decomposable functionality is a function with a subset of inputs that defines a strongly non-decomposable functionality. Since we consider the semi-honest model and so parties use only their prescribed inputs, it follows that the existence of a secure protocol for a non-decomposable function implies the existence of a secure protocol for the strongly non-decomposable function defined over the appropriate subset. It thus suffices to prove the theorem for a strongly non-decomposable function.

As we have described above, there are three steps in the proof of this theorem. In Sect. 3.3.1, we prove the first step. Specifically, in Lemma 3.9 we prove that there exists an "exclusive-revelation round" which is a round in which one party has learned something while the other has not, and then in Lemma 3.11 we prove that such a round must exist for inputs that form an insecure minor; we call this an "exclusive-revelation minor." Next in Sect. 3.3.2 we prove that the existence of an exclusive-revelation minor implies the existence of OT with weak correctness, and finally in Sect. 3.3.3 we explain how to boost the correctness and thus obtain full OT (with uniform infinitely often security).

3.3.1. *Step 1: The Existence of an Exclusive-Revelation Minor*

In order to prove our result, we exploit the fact that parties obtain information about the output of a computation gradually and that one party learns substantial information before the other party does. We begin with some notation regarding partial protocol executions.

For an r-round protocol π and a function $\nu : \mathbb{N} \to \mathbb{N}$ such that $\nu(n) \leq r(n)$ for all $n \in \mathbb{N}$, we denote by π_ν the protocol obtained by halting π after round $\nu(n)$ is completed. Specifically, the random variables $\text{VIEW}_{\mathsf{A}}^{\pi_\nu}(x, y, 1^n)$ and $\text{VIEW}_{\mathsf{B}}^{\pi_\nu}(x, y, 1^n)$ describe the views of A and B (respectively) in a random execution of π_ν on inputs (x, y) with security parameter n.

We next formally define what it means for a party to obtain non-trivial information about the other party's input.

Definition 3.7. (*distinguishing between inputs*) Let π be an r-round protocol for computing a functionality f (where r is some constant), and fix $i \in \mathbb{N}$. For a triple x, y, y' of inputs, we say that $\mathsf{A}(x)$ distinguishes between y and y' at round i if there exists a polynomial $p(\cdot)$ and a (uniform) PPT machine D such that for infinitely many n's,

$$\left|\Pr\left[D\left(\text{VIEW}_{\mathsf{A}}^{\pi_i}\left(x, y, 1^n\right), 1^n\right) = 1\right] - \Pr\left[D\left(\text{VIEW}_{\mathsf{A}}^{\pi_i}\left(x, y', 1^n\right), 1^n\right) = 1\right]\right| \geq \frac{1}{p(n)}$$

For a triple x, x', y of inputs we define that $\mathsf{B}(y)$ distinguishes between x and x' at round i in an analogous way.

As we will see below, it is crucial that D is a uniform PPT machine, since the parties need to run D in the OT protocol that we construct. Clearly, for any non-constant function, either A or B can distinguish (with some fixed input) between two distinct inputs for the other party at the last round of the protocol, for all but finitely many n's. The reason that we only require distinguishing for infinitely many n's is that we wish to use Definition 3.7 in order to identify the *first* round i that allows one of the parties to do so. This definition should, thus, be coupled with its converse, i.e., with the meaning of not being able to distinguish in any of the previous rounds. More intuition on that is given in the discussion at the end of Sect. 3.3.1.

We now define the notion of an exclusive-revelation round, which is just a round in which one party can distinguish inputs of the other, while the other cannot. Our formulation of this uses Definition 3.7.

Definition 3.8. (*exclusive-revelation round*) Let π be an r-round protocol for computing a symmetric functionality f. Then, π has an exclusive-revelation at round i if one of the following holds:

1. There exists a triple x, y, y' of inputs such that $\mathsf{A}(x)$ distinguishes between y and y' at round i, and there exists no triple $\hat{x}, \hat{x}', \hat{y}$ such that $\mathsf{B}(\hat{y})$ distinguishes between $\hat{x}$ and $\hat{x}'$ at round i (we say that x, y, y' define the revelation round); or
2. There exists a triple x, x', y of inputs such that $\mathsf{B}(y)$ distinguishes between x and x' at round i, and there exists no triple $\hat{x}, \hat{y}, \hat{y}'$ such that $\mathsf{A}(\hat{x})$ distinguishes between $\hat{y}$ and $\hat{y}'$ at round i (we say that x, x', y define the revelation round).

Protocol π has an exclusive-revelation round if there exists $0 \leq i \leq r$, such that π has an exclusive-revelation at round i.

We are now ready to prove that any constant-round protocol for computing a non-constant function (i.e., a function that has at least two different outputs) has an exclusive-revelation round.

Lemma 3.9. *Let f be a finite domain symmetric functionality that is non-constant. Let π be an r-round protocol for securely computing f (for some constant r). Then, π has an exclusive-revelation round.*

Proof. The intuition behind the proof of the lemma is as follows. Let π be a constant-round protocol for computing f. Then, on the one hand, before the first round of the protocol, none of the parties knows anything about the input of the other party. On the other hand, at the end of the protocol, both parties know the output of f. Hence for every triple x, x', y such that $f(x, y) \neq f(x', y)$, $\mathsf{B}(y)$ can distinguish between inputs x and x' for A; a similar argument holds for triples x, y, y' such that $f(x, y) \neq f(x, y')$. We deduce that there must exist a *first* round in which one of the parties can distinguish between two possible inputs of the other party and show that this round is an exclusive-revelation round.

Formally, for every (round number) $i \leq r$, every uniform PPT machine (distinguisher) D, and every triple x, x', y (recall that there are a finite number of such triples), we define

$$\begin{aligned}\varepsilon^{i,D}_{x,x',y}(n) = \big|&\Pr\left[D\left(\text{VIEW}^{\pi_i}_{\mathsf{B}}\left(x, y, 1^n\right), 1^n\right) = 1\right] \\ &- \Pr\left[D\left(\text{VIEW}^{\pi_i}_{\mathsf{B}}\left(x', y, 1^n\right), 1^n\right) = 1\right]\big|\end{aligned}$$

and let $r^D_{x,x',y}$ be the minimal round number $0 \leq i \leq r$ for which there exists a polynomial $p(\cdot)$ such that $\varepsilon^{i,D}_{x,x',y}(n) > \frac{1}{p(n)}$ for infinitely many n's. If no such i exists, we let $r^D_{x,x',y} = r + 1$. Note that this means that $r^D_{x,x',y}$ is the first round for which the PPT machine D can distinguish the ensembles $\left\{\text{VIEW}^{\pi_{r^D_{x,x',y}}}_{\mathsf{B}}(x, y, 1^n)\right\}_{n\in\mathbb{N}}$ and $\left\{\text{VIEW}^{\pi_{r^D_{x,x',y}}}_{\mathsf{B}}\left(x', y, 1^n\right)\right\}_{n\in\mathbb{N}}$.

We further define $r_{x,x',y} = \min_D\left\{r^D_{x,x',y}\right\}$ (this is well defined, as every $r^D_{x,x',y} \in [r+1]$). Observe that this means that $r_{x,x',y}$ is the minimal round for which there exists *any* uniform PPT machine that can distinguish the two ensembles (equivalently, the minimal round for which $\mathsf{B}(y)$ distinguishes between x and x'). For every triple x, y, y', we define $r_{x,y,y'}$ analogously.

By the correctness of the protocol, for every triple x, x', y such that $f(x, y) \neq f(x', y)$, the view of both parties after the last round (that is, round r) implies the output and hence there exists a uniform PPT machine D and a negligible function $\mu(\cdot)$ such that for all sufficiently large n's, $\varepsilon^{r,D}_{x,x',y}(n) \geq 1 - \mu(n)$. This in turn implies that for such triples, there exists a PPT machine D for which $r^D_{x,x',y} \leq r$, and hence $r_{x,x',y} \leq r$. Similarly, for every triple x, y, y' such that $f(x, y) \neq f(x, y')$, it holds that $r_{x,y,y'} \leq r$. Since f is not constant, there either exists a triple of the former type or of the latter type.

Now, define $i^*_{\mathsf{A}} = \min_{x,y,y'}\left\{r_{x,y,y'}\right\}$ and $i^*_{\mathsf{B}} = \min_{x,x',y}\left\{r_{x,x',y}\right\}$. Note that i^*_{A} is the minimal round for which there exists a triple x, y, y' such that $\mathsf{A}(x)$ distinguishes between y and y', and i^*_{B} is the minimal round for which there exists a triple x, x', y such that $\mathsf{B}(y)$ distinguishes between x and x'. Since f is not constant, it holds that either $i^*_{\mathsf{A}} \leq r$ or $i^*_{\mathsf{B}} \leq r$ (or both). We claim that π has exclusive-revelation either at round i^*_{A} or at round i^*_{B}.

Clearly, the view of a party does not change in the round that it is active, and hence, neither does its distinguishing capability. Hence, i^*_{A} must be even and i^*_{B} must be odd (recall that we assume that A sends the first message). It follows that $i^*_{\mathsf{A}} \neq i^*_{\mathsf{B}}$. Assume that $i^*_{\mathsf{A}} < i^*_{\mathsf{B}}$. By the definition of i^*_{B}, we know that there exists no triple $\hat{x}, \hat{x}', \hat{y}$ such that $\mathsf{B}(\hat{y})$ distinguishes between $\hat{x}$ and $\hat{x}'$ at round i^*_{A}. Hence, there is a revelation round in π at round $i^*_{\mathsf{A}} \leq r$. The case of $i^*_{\mathsf{A}} > i^*_{\mathsf{B}}$ is analogous. □

We complete this step of the proof of Theorem 3.5 by showing that when a strongly non-decomposable function has a protocol with an exclusive-revelation round, this round is defined by inputs that form an insecure minor (see Definition 3.1).

Definition 3.10. (*exclusive-revelation minor*) Let π be a protocol for computing a symmetric functionality f. If there exists an X-minor x, x', y, y' with respect to f such that x', y, y' define an exclusive-revelation round for π, then we say that π has an exclusive-revelation X-minor; an exclusive-revelation Y-minor is defined analogously. We say that π has an exclusive-revelation minor if it has an exclusive-revelation X-minor or an exclusive-revelation Y-minor.

The next lemma states that strongly non-decomposable functions have the property that the existence of an exclusive-revelation round implies the existence of an exclusive-revelation minor.

Lemma 3.11. *Let π be a protocol that securely computes a strongly non-decomposable symmetric function f with a finite domain. If π has an exclusive-revelation round then it has an exclusive-revelation minor.*

Proof. The proof follows by analyzing the general structure of strongly non-decomposable functions. Let f be *any* symmetric strongly non-decomposable function with a finite domain. Assume that there exist x_j, y_k, y_ℓ (with $k < \ell$) that define an exclusive-revelation at round i; that is, $\mathsf{A}(x_j)$ distinguishes between y_k and y_ℓ at round i. We show that this implies that π has an exclusive-revelation X-minor. Since f is a strongly non-decomposable function, it holds that $y_k \equiv y_\ell$. Let $y_{i_1}, \ldots, y_{i_t}$ be such that $y_k \sim y_{i_1} \sim \ldots \sim y_{i_t} \sim y_\ell$ and let $y_{i_0} = y_k$ and $y_{i_{t+1}} = y_\ell$. $\mathsf{A}(x_j)$ distinguishes between y_{i_0} and $y_{i_{t+1}}$ at round i, and since t is a constant (recall that f has a finite domain), there exists some $h \in [t+1]$ such that $\mathsf{A}(x_j)$ distinguishes between $y_{i_{h-1}}$ and y_{i_h} at round i. Now, by definition, since $y_{i_{h-1}} \sim y_{i_h}$, there exists some x such that $f(x, y_{i_{h-1}}) = f(x, y_{i_h})$. Hence, $x, x_j, y_{i_{h-1}}, y_{i_h}$ forms an exclusive-revelation X-minor. The proof for the case that B distinguishes is analogous. □

Infinitely Often. Observe that the existence of an exclusive-revelation minor means that there exists an insecure minor and a round of the protocol such that one party can distinguish the other party's inputs at this round while the other cannot. We stress that a party *distinguishes inputs* if it has polynomial advantage in guessing the input *for infinitely many n's*. It would be preferable to prove this for all sufficiently large n's, since this would enable us to later construct a fully secure oblivious transfer protocol, and not just an infinitely often secure oblivious transfer protocol. However, we are unable to do this since we need to utilize the existence of a round where one party has learned something and the other has not learned anything. We prove this by taking the first such round, and this guarantees that in any previous round the other party has not learned anything, except possibly for a finite number of n's. This means that it did not learn for all but a finite number of the n's in which the other party did learn, as required. In contrast, if we were to take the first round in which one party learns for all sufficiently large n's, then it is possible that the other party has learned for infinitely many of these n's in a previous round, and so security will not be guaranteed.

Constant-Round. We use the assumption that π is constant-round in the proof that π has an exclusive-revelation round (Lemma 3.9). Recall that an exclusive-revelation round is the *first* round that a party can distinguish between the inputs of the other party *for*

infinitely many n's. If the number of rounds in π is non-constant, then for every n the concrete number of rounds in the protocol is different and hence we would have to define an "exclusive-revelation function"; that is, a function $\nu : \mathbb{N} \to$ *round number*, that defines the first round (as a function of n) that a party can distinguish between the inputs of the other party. It is not clear how to define such a function, and moreover, how to prove its existence.

Finite Domain. We restrict ourselves to functions with finite domains (i.e., not dependent on the security parameter) in order to be consistent with previous works studying completeness and triviality of symmetric functions ([12,17]). Extending the study of completeness to functions with non-finite domains is beyond the scope of this paper.

3.3.2. *Step 2: From an Exclusive-Revelation Minor to io-Weak-OT*

We now show that if a function has a protocol with an exclusive-revelation minor, then it can be used to obtain a weak version of oblivious transfer. The "weakness" in the OT is with respect to correctness, and not privacy. Below we provide a formal definition for uniform infinitely often weak oblivious transfer. The definition is based on the definition of uniform infinitely often security (see Definition 2.3).

Definition 3.12. A protocol π is a uniform infinitely often weak oblivious transfer protocol (io-weak-OT) if there exists an infinite set $\mathcal{N} \subseteq \mathbb{N}$ such that

Correctness: There exists a polynomial $p(\cdot)$ such that for every $n \in \mathbb{N}$ and every pair of inputs $b, c \in \{0,1\}$, it holds that

$$\Pr\left[\text{Output}^{\pi}(b, c, 1^n) = \text{OT}(b, c)\right] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

where the probability is taken over the random coins of the parties.

Privacy: The same as in Definition 2.3.

We stress that the privacy requirement of the oblivious transfer is identical to uniform infinitely often security in Definition 2.3, while the correctness requirement is weaker since it is only required that correctness holds with probability noticeably greater than $1/2$, and not close to 1.

Lemma 3.13. *Let $\pi = (\mathsf{A}, \mathsf{B})$ be a PPT protocol for securely computing a functionality f. If π has an exclusive-revelation minor, then there exists a PPT protocol $\widetilde{\pi}$ that is a uniform infinitely often weak oblivious transfer.*

Proof. Intuitively, the existence of an exclusive-revelation round in the protocol allows us (in some weak sense) to move to the realm of asymmetric functionalities where one party learns the output, while the other party learns nothing. It is known that an asymmetric functionality containing an insecure minor implies OT. We therefore use the insecure minor guaranteed by the hypothesis of the lemma to construct (a weak form of) OT in a way similar to that used in the world of asymmetric computation. The formal arguments follow.

Let π be a protocol computing a symmetric functionality f. Assume without loss of generality that there exists an X-minor x, x', y, y' with respect to f for π (the case of an exclusive-revelation Y-minor is analogous). Assume that x', y, y' define an exclusive-revelation at round i. Specifically, for *every* triple $\hat{x}, \hat{x}', \hat{y}$, we have that $\mathsf{B}(\hat{y})$ does not distinguish between $\hat{x}$ and $\hat{x}'$ at round i. Let D be the corresponding distinguisher, and assume without loss of generality that it always outputs either 0 or 1. Since $f(x, y) = f(x, y')$ (by definition of a minor), by the security of π we also have that $\mathsf{A}(x)$ does not distinguish between y and y' at round i (or any round, for that matter). It is without loss of generality (e.g., by interchanging y and y') to assume that for infinitely many n's that

$$\Pr\left[D\left(\text{VIEW}_{\mathsf{A}}^{\pi_i}\left(x', y, 1^n\right), 1^n\right) = 1\right] - \Pr\left[D\left(\text{VIEW}_{\mathsf{A}}^{\pi_i}\left(x', y', 1^n\right), 1^n\right) = 1\right] \geq \frac{1}{p(n)} \tag{4}$$

We next construct an io-weak-OT protocol $\widetilde{\pi}$. The idea is to run the protocol on the inputs of the above minor until round i and then to halt the execution. As B learns nothing from the computation, we can let the sender $\widetilde{\mathsf{Sen}}$ play the role of B. The receiver $\widetilde{\mathsf{Rec}}$ plays the role of A using x as its input for π, if $c = 0$ (hence, learning nothing), and using x' as its input for π if $c = 1$.

Regarding the sender's input, it may seem natural to have $\widetilde{\mathsf{Sen}}$ use y' in case $b = 0$ and y in case $b = 1$ (the receiver would then output whatever the distinguisher outputs). However, it is possible that the distinguisher outputs 0 with probability $3/4$ on input (x', y), and with probability $3/4 + 1/p(n)$ on input (x', y'). In such a case, the receiver will output 0 with probability $3/4$ even when the output is supposed to be 1, and so weak correctness will not hold (recall that we need correctness with probability greater than $1/2$). To overcome this, we have the sender use a random input in $\{y, y'\}$ and therefore transfer a random bit r to the receiver (who in turn will try to learn r only if its input is $c = 1$). The sender then sends the receiver the bit $z = r \oplus c$, and the receiver outputs z if the distinguisher output 0 and $z \oplus 1$ otherwise. This has the effect of moving the error to be around $1/2$, and so we obtain correctness $1/2 + 1/p(n)$.

Protocol 3.14. (An io-weak-*OT* $\widetilde{\pi} = \left(\widetilde{\mathsf{Sen}}, \widetilde{\mathsf{Rec}}\right)$)

Inputs: *The private input of the sender* $\widetilde{\mathsf{Sen}}$ *is a bit* $b \in \{0, 1\}$*, and the private input of the receiver* $\widetilde{\mathsf{Rec}}$ *is a bit* $c \in \{0, 1\}$*. The common input is* 1^n*, where n is the security parameter.*

The protocol:

1. *The sender chooses a random bit* $r \in \{0, 1\}$.
2. *The parties start an execution of* π*, where the sender* $\widetilde{\mathsf{Sen}}$ *plays the role of* B *and the receiver* $\widetilde{\mathsf{Rec}}$ *plays the role of* A*. The inputs of the parties are set as follows:*
   - *The input of* B *(played by* $\widetilde{\mathsf{Sen}}$*) is* y' *if* $r = 0$ *and* y *if* $r = 1$.
   - *The input of* A *(played by* $\widetilde{\mathsf{Rec}}$*) is* x *if* $c = 0$ *and* x' *if* $c = 1$.

The parties halt after the ith round of π. Let v^i_{A} be the partial view of A in this partial execution of π.

3. *The sender $\widetilde{\mathsf{Sen}}$ sends $z = r \oplus b$ to the receiver $\widetilde{\mathsf{Rec}}$.*
4. *If $c = 0$, the receiver outputs λ. Otherwise (if $c = 1$), the receiver runs D on v^i_{A} to get its output, denoted r', and outputs $z \oplus r'$. The sender always outputs λ.*

Note that the receiver is able to run the distinguisher D since D is a uniform Turing machine.

Proving the Weak Correctness of the Protocol. Proving the correctness when $c = 0$ is trivial since both parties will always output λ as required. We consider the case that $c = 1$. We need to show that there exists a polynomial $q(\cdot)$ such that for infinitely many n's, it holds that $\Pr\left[\text{Output}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}(b, c = 1, 1^n) = b\right] \geq \frac{1}{2} + \frac{1}{q(n)}$. We will show that this holds for the polynomial $q(n) = 2p(n)$ and for all n's for which Eq. (4) is satisfied. We fix such an n.

Recall that $\widetilde{\mathsf{Rec}}$ outputs $z \oplus r'$, where $z = b \oplus r$ and hence the output of $\widetilde{\mathsf{Rec}}$ equals b if and only if $r' = r$, where r' denotes the output of D on the partial view v^i_{A}. Thus, it suffices to give a lower bound on the following term (recall that we consider the case that $\widetilde{\mathsf{Rec}}$ uses x' since $c = 1$):

$$
\begin{aligned}
&\Pr\left[r' = r\right] \\
&= \Pr[r = 0] \cdot \Pr\left[r' = 0 \mid r = 0\right] + \Pr[r = 1] \cdot \Pr\left[r' = 1 \mid r = 1\right] \\
&= \frac{1}{2} \cdot \Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y', 1^n\right), 1^n\right) = 0\right] + \frac{1}{2} \cdot \Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y, 1^n\right), 1^n\right) = 1\right] \\
&= \frac{1}{2} \cdot \left(1 - \Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y', 1^n\right), 1^n\right) = 1\right]\right) + \frac{1}{2} \cdot \Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y, 1^n\right), 1^n\right) = 1\right] \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y, 1^n\right), 1^n\right) = 1\right] - \Pr\left[D\left(\text{View}^{\pi_i}_{\mathsf{A}}\left(x', y', 1^n\right), 1^n\right) = 1\right]\right) \\
&\geq \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{p(n)} \qquad (5)
\end{aligned}
$$

where the last inequality is from Eq. (4).

Proving the Privacy of the Protocol. We now proceed to proving that Eq. (2) and (3) in Definition 2.3 hold for all sufficiently large n's (and thus, in particular, for infinitely many n's for which weak correctness holds, as required in Definition 2.3).

Simulating the View of the Sender. We construct a PPT machine $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ that simulates the sender's view. $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ receives as input the sender's input b and the security parameter 1^n, and works as follows:

1. $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ chooses a random bit $r_{\widetilde{\mathsf{Sen}}} \in \{0, 1\}$.
2. $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ then starts an execution of π on the following inputs until the ith round:
   - If $r_{\widetilde{\mathsf{Sen}}} = 0$, the input of B is y' and if $r_{\widetilde{\mathsf{Sen}}} = 1$, the input of B is y.
   - The input of A is x.
3. $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ outputs $r_{\widetilde{\mathsf{Sen}}}$ and the partial view v^i_{B} of B.

We next prove that for every pair of inputs $b, c \in \{0,1\}$,

$$\left\{\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n)\right\}_{n\in\mathbb{N}} \stackrel{C}{\equiv}_U \left\{\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n)\right\}_{n\in\mathbb{N}}.$$

Assume to the contrary that these two random variables can be distinguished by a uniform machine, that is, there exist $b, c \in \{0,1\}$, a polynomial $p'(\cdot)$, and a (uniform) PPT machine D' such that for infinitely many n,

$$\left|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1\right]-\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1\right]\right| \geq \frac{1}{p'(n)}. \tag{6}$$

Both of the above random variables describe a random view for the sender $\widetilde{\mathsf{Sen}}$ in an execution of $\widetilde{\pi}$, with the same value of b. The only difference between an output of $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ and the view of the sender in a real execution of $\widetilde{\pi}$ is that $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ *always* uses x as the input of A (put differently, acts as if $c=0$), whereas in a real execution of $\widetilde{\pi}$, the input of A is set according to the value of c (equals x if $c=0$ and equals x' if $c=1$). Thus, Eq. (6) can only ever hold when $c=1$, since for $c=0$ the above random variables are identical. Hence, in the following, we fix $c=1$.

We consider the success probability of the distinguisher D' based on the values of the random bit r (in a random execution of π) and $r_{\widetilde{\mathsf{Sen}}}$ (in a simulation). Since both are selected uniformly at random, we have that:

$$\begin{aligned}
&\left|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1\right]-\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1\right]\right| \\
&= \Bigg|\frac{1}{2}\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=0\right] \\
&\quad+\frac{1}{2}\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=1\right] \\
&\quad-\frac{1}{2}\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=0\right] \\
&\quad-\frac{1}{2}\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=1\right]\Bigg| \\
&\leq \frac{1}{2}\Big|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=0\right] \\
&\quad-\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=0\right]\Big| \\
&\quad+\frac{1}{2}\Big|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=1\right] \\
&\quad-\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=1\right]\Big|.
\end{aligned}$$

Hence, the distinguishing gap of D' is at least $\frac{1}{p'(n)}$ (for infinitely many n's) either when we condition on $r = r_{\widetilde{\mathsf{Sen}}} = 0$ or when we condition on $r = r_{\widetilde{\mathsf{Sen}}} = 1$. Assume without loss of generality that the former holds. That is, for infinitely many n's it holds that

$$\left|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=0\right]\right.$$
$$\left.-\Pr\left[D'\left(\mathrm{VIEW}^{\tilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=0\right]\right| \geq \frac{1}{p'(n)}. \tag{7}$$

We construct a distinguisher D'' for $\left\{\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x,y',1^n\right)\right\}_{n\in\mathbb{N}}$ and $\left\{\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x',y',1^n\right)\right\}_{n\in\mathbb{N}}$. On a view v, the distinguisher D'' will apply D' to $(0,v)$ and return the same output. By definition of Protocol 3.14, we have that for every n the view of $\widetilde{\mathsf{Sen}}$ conditioned on $r=0$ (and $c=1$) is identically distributed to $\left(0,\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x',y',1^n\right)\right)$. By the definition of the simulator $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$, we have that for every n the output of $\mathcal{S}_{\widetilde{\mathsf{Sen}}}$ conditioned on $r_{\widetilde{\mathsf{Sen}}}=0$ is identically distributed to $\left(0,\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x,y',1^n\right)\right)$. It follows that

$$\begin{aligned}&\left|\Pr\left[D''\left(\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x,y',1^n\right),1^n\right)=1\right]-\Pr\left[D''\left(\mathrm{VIEW}^{\pi_i}_{\mathsf{B}}\left(x',y',1^n\right),1^n\right)=1\right]\right|\\&=\left|\Pr\left[D'\left(\mathcal{S}_{\widetilde{\mathsf{Sen}}}(b,\lambda,1^n),1^n\right)=1 \mid r_{\widetilde{\mathsf{Sen}}}=0\right]\right.\\&\quad\left.-\Pr\left[D'\left(\mathrm{VIEW}^{\tilde{\pi}}_{\widetilde{\mathsf{Sen}}}(b,c,1^n),1^n\right)=1 \mid r=0\right]\right|.\end{aligned}$$

By Eq. (7) the latter term is at least $\frac{1}{p'(n)}$ for infinitely many n's. This means that $\mathsf{B}(y')$ distinguishes between x and x' at round i, which is a contradiction to the assumption that x', y, y' define an exclusive-revelation at round i for π.

Simulating the View of the Receiver. In the case that $c=1$ the simulator receives both the sender's and receiver's inputs c and b, and hence, can perfectly simulate the view of the receiver by just running the protocol on these inputs. We, therefore, describe the simulator only for the case that $c=0$. The simulator $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ receives as input the bit $c=0$, the output $\mathrm{OT}_R=\lambda$ of the functionality OT to the receiver, and the security parameter 1^n, and works as follows:

1. $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ executes π for i rounds, running A with input x and B with input y.
2. $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ chooses a random bit $z_{\mathcal{S}}\in\{0,1\}$.
3. $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ outputs $z_{\mathcal{S}}$ appended to the partial view v^i_{A} of A.

The difference between the simulated view and a real view is that in a real execution, the sender playing B sometimes uses y and sometimes uses y', whereas in the simulated execution it always uses y. In addition, the simulator sends a random $z_{\mathcal{S}}$ that is not correlated to the value r implied by the input used by B in the computation of π. However, since $f(x,y)=f(x,y')$ the receiver (using x as input) learns nothing about whether the sender used y or y', or about the correlation between z and r.

Formally, we show that for every $b\in\{0,1\}$,

$$\left\{\mathcal{S}_{\widetilde{\mathsf{Rec}}}(c=0,\lambda,1^n)\right\}_{n\in\mathbb{N}} \overset{C}{\equiv}_U \left\{\mathrm{VIEW}^{\tilde{\pi}}_{\widetilde{\mathsf{Rec}}}(b,c=0,1^n)\right\}_{n\in\mathbb{N}}.$$

To prove this, we consider a hybrid simulator $\mathcal{S}^h$ that also works for the case that $c=0$ but gets b as input nevertheless. $\mathcal{S}^h$ works as follows:

1. $\mathcal{S}^h$ chooses a random bit $r_{\mathcal{S}^h}\in\{0,1\}$.

2. $\mathcal{S}^h$ starts an execution of π on the following inputs until the ith round:
   - The input of A is x, and the input of B is (always) y.
3. $\mathcal{S}^h$ sets $z_{\mathcal{S}^h} = b \oplus r_{\mathcal{S}^h}$.
4. $\mathcal{S}^h$ outputs $z_{\mathcal{S}^h}$ appended to the partial view v^i_{A} of A.

It suffices to show that for every b it holds that,

$$\left\{\mathcal{S}_{\widetilde{\mathsf{Rec}}}(c=0,\lambda,1^n)\right\}_{n\in\mathbb{N}} \overset{C}{\equiv}_U \left\{\mathcal{S}^h\left(c=0,b,1^n\right)\right\}_{n\in\mathbb{N}} \tag{8}$$

and

$$\left\{\mathcal{S}^h\left(c=0,b,1^n\right)\right\}_{n\in\mathbb{N}} \overset{C}{\equiv}_U \left\{\textsc{View}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}\left(b,c=0,1^n\right)\right\}_{n\in\mathbb{N}}. \tag{9}$$

It is easy to observe that the only difference between $\mathcal{S}^h$ and $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ is that $\mathcal{S}^h$ sets $z_{\mathcal{S}^h} = b \oplus r_{\mathcal{S}^h}$ and $\mathcal{S}_{\widetilde{\mathsf{Rec}}}$ lets $z_{\mathcal{S}}$ be a random bit. However, for every b both $z_{\mathcal{S}^h}$ and $z_{\mathcal{S}}$ are uniform bits and, furthermore, are independent of everything else in the simulation. Hence,

$$\left\{\mathcal{S}_{\widetilde{\mathsf{Rec}}}(c=0,\lambda,1^n)\right\}_{n\in\mathbb{N}} \equiv \left\{\mathcal{S}^h\left(c=0,b,1^n\right)\right\}_{n\in\mathbb{N}},$$

and Eq. (8) follows.

We prove that Eq. (9) also holds by contradiction. Assume to the contrary that there exists $b \in \{0,1\}$, a uniform PPT machine D', and a polynomial $p'(\cdot)$ such that for infinitely many n's,

$$\left|\Pr\left[D'\left(\mathcal{S}^h\left(c=0,b,1^n\right)\right)=1\right] - \Pr\left[D'\left(\textsc{View}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}\left(b,c=0,1^n\right)\right)=1\right]\right| \geq \frac{1}{p'(n)} \tag{10}$$

Since z (in an execution of $\widetilde{\pi}$) and $z_{\mathcal{S}^h}$ are identically distributed given c and b (which are now fixed), the only place where the two distribution differ is the partial view v^i_{A}: in $\mathcal{S}^h\left(c=0,b,1^n\right)$, this partial view is generated by executing π on inputs x and y, whereas in $\textsc{View}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}(b,c=0,1^n)$, this partial view is obtained by executing π on inputs x and y_r where $y_r = y$ if $r=1$ and $y_r = y'$ if $r=0$. Hence, when $r=1$, the two distributions are identically distributed. Since $\Pr[r=0] = \Pr\left[r_{\mathcal{S}^h}=0\right] = \frac{1}{2}$, it follows that

$$\begin{aligned}\frac{1}{p'(n)} &\leq \left|\Pr\left[D'\left(\mathcal{S}^h\left(c=0,b,1^n\right)\right)=1\right] - \Pr\left[D'\left(\textsc{View}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}\left(b,c=0,1^n\right)\right)=1\right]\right| \\ &\leq \frac{1}{2}\left|\Pr\left[D'\left(\mathcal{S}^h\left(c=0,b,1^n\right)\right)=1 \mid r_{\mathcal{S}^h}=0\right]\right. \\ &\quad \left. - \Pr\left[D'\left(\textsc{View}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}\left(b,c=0,1^n\right)\right)=1 \mid r=0\right]\right|, \end{aligned} \tag{11}$$

and hence,

$$\left|\Pr\left[D'\left(\mathcal{S}^h\left(c=0,b,1^n\right)\right)=1 \mid r_{\mathcal{S}^h}=0\right]\right. \\ \left.-\Pr\left[D'\left(\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}\left(b,c=0,1^n\right)\right)=1 \mid r=0\right]\right| \geq \frac{2}{p'(n)}. \tag{12}$$

We use D' construct a uniform PPT machine D'' such that for infinitely many n's

$$\left|\Pr\left[D''\left(\mathrm{VIEW}^{\pi_i}_{\mathsf{A}}\left(x,y,1^n\right)\right)=1\right]-\Pr\left[D''\left(\mathrm{VIEW}^{\pi_i}_{\mathsf{A}}\left(x,y',1^n\right)\right)=1\right]\right| \geq \frac{2}{p'(n)}, \tag{13}$$

which is a contradiction to the security of π.

We conclude the proof by describing D'' (that knows the bit b for which Eq. (10) holds). On a view v^i_{A} that is either generated in a partial execution of $\pi(x,y)$ until the ith round or in an execution of $\pi(x,y')$ until the ith round, D'' sets $z=b\oplus 0=b$ and $v':=\left(v^i_{\mathsf{A}},z\right)$ and applies D' to v' (and outputs whatever D' outputs). If v^i_{A} was generated by $\pi(x,y)$, then v' is distributed as $\left\{\mathcal{S}^h\left(c=0,b,1^n\right) \mid r_{\mathcal{S}^h}=0\right\}$ and if v^i_{A} was generated in an execution of $\pi(x,y')$, then v' is distributed as $\left\{\mathrm{VIEW}^{\widetilde{\pi}}_{\widetilde{\mathsf{Rec}}}(b,c=0,1^n) \mid r=0\right\}$. Hence, Eq. (13) follows, which concludes the proof. □

Uniform Security. As explained above, the privacy of the receiver is preserved by the exclusiveness of the revelation minor (in round i). That is, since the sender in the OT protocol takes the role of the party that cannot distinguish the inputs of the other party (the one active in round i). By Definition 3.7, no *uniform* distinguisher D succeeds with non-negligible probability in distinguishing the two possible inputs of the receiver. It does not, however, rule out the possibility that a *non-uniform* distinguisher has noticeable success probability, yielding the privacy of the receiver vulnerable with respect to non-uniform adversaries.

3.3.3. *From Weak Uniform io-OT to Uniform io-OT*

We conclude the proof by arguing that the existence of a uniform infinitely often weak-OT implies the existence of a uniform infinitely often OT protocol. Let π be a uniform infinitely often weak-OT protocol. We construct a uniform infinitely often OT protocol $\widetilde{\pi}$ by having the parties run polynomially many executions of π on their inputs. If $c=1$, the receiver outputs the majority of the outputs of the receiver in π, and otherwise it outputs λ. It follows from the Chernoff bound that for the infinitely many n's for which π has weak correctness, $\widetilde{\pi}$ is correct with probability $1-\mu(n)$, for some negligible function $\mu(\cdot)$. To prove the privacy of $\widetilde{\pi}$, we use multiple executions of the simulators of the io-weak-OT. A standard hybrid argument shows that this yields a satisfactory simulation for the io-OT protocol. We stress that a simple hybrid argument works because the parties are semi-honest and hence follow the prescribed protocol (specifically, they select fresh random coins for each execution).

This completes the proof of Theorem 3.5.

4. Black-Box and Existential Completeness

4.1. *Black-Box Reduction and Completeness*

Loosely speaking, a functionality is called complete if it can be used to securely compute any functionality. In the standard definitions of completeness used in previous works (cf. [2,12,17]), this is defined via the notion of "reduction." Specifically, g reduces to f if it is possible to securely compute g *given access to* f, and a functionality is complete if all functionalities reduce to it. In this section, we explore in greater depth how this notion of reduction is defined and what the ramifications of this definition are.

The definition of reduction in most previous works uses the notion of an *ideal black-box* for computing a functionality $f = (f_A, f_B)$. The parties A and B run a protocol for computing g while given access to an incorruptible trusted party who computes f for them throughout the execution (the parties send inputs x and y to the trusted party, which computes $f(x, y) = (f_A(x, y), f_B(x, y))$, and sends them back their respective outputs). A functionality g reduces to a functionality f, if g is securely computable given such a trusted party for computing f. This notion is equivalent to the notion of oracle-aided protocols, defined in [6, Section 7.3.1]. Formally, using the terminology of [6], black-box reductions can be defined as follows.

Definition 4.1. (*black -box reductions*) Let g and f be two functionalities. We say that g black-box reduces to f if there exists an oracle-aided protocol π that *information-theoretically* securely computes g when using the oracle functionality f.

Definition 4.1 is the definition used by most previous works, with two exceptions being the works of Beimel et al. [2] and Harnik et al. [8]. Both these works considered computational security rather than information-theoretic, where [8] still considered black-box reductions, while [2] considered non-black-box reductions. We now proceed with the definitions of *black-box-complete functions* and *trivial functions*.

Definition 4.2. (*black-box-complete*) A functionality f is called black-box-complete if all g black-box-reduce to it.

Definition 4.3. A functionality f is called trivial if it can be information-theoretically securely computed with no oracle.

The picture of completeness and triviality according to the above definitions is well known for both the case of asymmetric functionalities where only one of the parties receives output and the case of symmetric functionalities where the parties receive the same output (i.e., $f_A = f_B$). Specifically:

Theorem 4.4. ([2,12,17]) *An asymmetric functionality is black-box-complete if it contains an insecure minor, and trivial if not. Furthermore, a symmetric functionality is black-box-complete if and only if it contains an embedded OR, and is trivial if and only if it is decomposable (see Definition 3.4).*

Combining the above with Theorem 3.5, we have the following corollary:

Corollary 4.5. *There exist symmetric deterministic functionalities over a domain of constant-size that are neither trivial nor black-box-complete, such that if there exists a constant-round protocol π that securely computes such a function, then there exists a uniform infinitely often OT protocol.*

We remark that using the results of Kilian [11], one can show that any functionality can be securely computed with uniform infinitely often security (Definition 2.3) given a uniform infinitely often OT protocol. It therefore seems unlikely that such an OT protocol can be constructed under weaker assumption than fully secure OT (at least, infinitely often secure protocols are not known to be constructible under weaker assumptions, and the known black-box separations for OT [5,9] hold also for infinitely often OT).

4.2. *Existential Completeness: An Alternative Formulation*

Corollary 4.5 suggests that there may exist functionalities that are neither trivial nor complete and yet are in some sense complete (albeit, under the caveat of uniform infinitely often security). This is due to the fact that the definition of black-box-completeness relates to the computation of f as *atomic*, whereas in real life, computation is carried out step-by-step and, in particular, is *not* black-box in the functionality. We therefore present an alternative notion of completeness which is purely *existential*. Informally, our definition is based on saying that f "implies" g in some sense if the feasibility of securely computing g is implied by the feasibility of securely computing f. Formally:

Definition 4.6. Let $\mathcal{U}$ denote the set of all polynomial time computable functionalities. The achievable class of $f \in \mathcal{U}$, denoted as $\mathsf{C}(f)$, is the set of all $g \in \mathcal{U}$ such that if there exists a computationally secure protocol π_f for computing f, then there exists a computationally secure protocol π_g for computing g.

Let $f, g \in \mathcal{U}$. We say that g existentially reduces to f if $g \in \mathsf{C}(f)$. Functionality f is existentially trivial if $f \in \mathsf{C}(f_\lambda)$ (where $f_\lambda(\cdot,\cdot) = (\lambda, \lambda)$), and is existentially complete if $\mathsf{C}(f) = \mathcal{U}$.

The above definition follows the intuition that a functionality is trivial if it can be securely computed "with no help," and complete if all functionalities can be securely computed if it can be securely computed. We stress that if enhanced trapdoor functions exist, then all functionalities are trivial and complete by this definition. Nevertheless, our definition is helpful since a proof that a functionality f is complete (without proving the existence of enhanced trapdoor permutations) is essentially a proof that f requires an assumption that implies OT. We remark that this is the same as in the definition of (black-box) computational completeness that appears in [8]. We also note that any functionality that is black-box-complete, or complete by the computational definition in [8], is also existentially complete.

Evidently, a black-box notion of reduction is an elegant one. Indeed, this notion is the one used in [8], where a function g is said to be reducible to f, if it is possible to construct a computationally secure protocol for g given a black-box for computing f. However, such a notion does not allow for a construction of a secure protocol for g that depends on the actual messages of the secure protocol for computing f. Specifically,

a black-box construction of a protocol for g cannot run an r-round protocol π for f for $i < r$ rounds (as done in this work) and cannot contain zero-knowledge proofs that parties follow the protocol π, as done in the work of Beimel et al. [2], which employs the compiler of [7].

We conclude by remarking that the definition of existential completeness (Definition 4.6) has the advantage that it can more accurately map the assumptions required for securely computing a functionality. In particular, a function that is not complete cannot imply OT, something which *can* happen under the black-box definition (as hinted to by Corollary 4.5). However, it is also true that the definition of existential completeness is less helpful due to its non-constructive nature. Specifically, it does not enable us to prove or consider a hierarchy of functionalities, and a proof that $g \in \mathsf{C}(f)$ does not necessarily tell us how to securely compute g, even given a protocol for securely computing f.

Acknowledgements

We are grateful to Amos Beimel for helpful discussions. We thank Daniel Kraschewski for helping us understand the work of [16]. We also thank the anonymous referees of the Journal of Cryptology for their useful comments, and specifically, for pointing out Remark 3.6 to us.

References

[1] D. Beaver, Perfect privacy for two-party protocols, in *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography* (1989)

[2] A. Beimel, T. Malkin, S. Micali, The all-or-nothing nature of two-party secure computation, in *Advances in Cryptology—CRYPTO '99*, ed By M. Wiener. Lecture Notes in Computer Science, vol. 1666 (Springer, 1999), pp. 80–97

[3] B. Chor, E. Kushilevitz. A zero-one law for Boolean privacy. *SIAM J. on Discrete Mathematics*, **4**(1), 36–47, (1991)

[4] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts. *CACM*, **28**(6), 637–647 (1985)

[5] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, M. Viswanathan, The relationship between public key encryption and oblivious transfer, in *Proc. of the 41st IEEE Symp. on Foundations of Computer Science* (2000), pp. 325–335

[6] O. Goldreich. *Foundations of Cryptography, Voume II – Basic Applications*. (Cambridge University Press, 2004)

[7] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game, in *Proc. of the 19th ACM Symp. on the Theory of Computing* (1987), pp. 218–229

[8] D. Harnik, M. Naor, O. Reingold, A. Rosen, Completeness in two-party secure computation: a computational view, in *Proc. of the 36th ACM Symp. on the Theory of Computing* (2004), pp. 252 – 261

[9] R. Impagliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in *Proc. of the 21st ACM Symp. on the Theory of Computing* (1989), pp. 44–61

[10] R.A. Jeffs, M. Rosulek, Characterizing the cryptographic properties of reactive 2-party functionalities, in *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*. TCC'13 (Springer, Berlin, 2013), pp. 263–280

[11] J. Kilian, Founding cryptography on oblivious transfer, in *Proc. of the 20th ACM Symp. on the Theory of Computing* (1988), pp. 20–31

[12] J. Kilian, A general completeness theorem for two-party games, in *Proc. of the 23th ACM Symp. on the Theory of Computing* (1991), pp. 553–560
[13] J. Kilian, More general completeness theorems for two-party games, in *Proc. of the 32nd ACM Symp. on the Theory of Computing* (2000), pp. 316–324
[14] J. Kilian, E. Kushilevitz, S. Micali, R. Ostrovsky, Reducibility and completeness in private computations. *SIAM J. Comput.* **28**(4), 1189–1208, (2000). This is the journal version of [12, 18]
[15] D. Kraschewski, H.K. Maji, M. Prabhakaran, A. Sahai, A full characterization of completeness for two-party randomized function evaluation, in *Advances in Cryptology—EUROCRYPT 2014—33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11–15, 2014. Proceedings* (2014), pp. 659–676
[16] D. Kraschewski, J. Müller-Quade, Completeness theorems with constructive proofs for finite deterministic 2-party functions, in *Proceedings of the 8th Conference on Theory of Cryptography*, TCC'11 (Springer, Berlin, 2011) pp. 64–381
[17] E. Kushilevitz, Privacy and communication complexity. *SIAM J. on Discrete Mathematics*, **5**(2), 273–284 (1992)
[18] E. Kushilevitz, S. Micali, R. Ostrovsky, Reducibility and completeness in multi-party private computations, in *Proc. of the 35th IEEE Symp. on Foundations of Computer Science* (1994), pp. 478–491
[19] Y. Lindell, E. Omri, H. Zarosim, Completeness for symmetric two-party functionalities—revisited, in *Advances in Cryptology—ASIACRYPT 2012* (2012), pp. 116–133
[20] H.K. Maji, M. Prabhakaran, M. Rosulek, A unified characterization of completeness and triviality for secure function evaluation, in *INDOCRYPT* (2012), pp. 40–59
[21] H.K. Maji, M. Prabhakaran, M. Rosulek, Complexity of multi-party computation problems: the case of 2-party symmetric secure function evaluation, in *Proc. of the Sixth Theory of Cryptography Conference—TCC 2009* (2009), pp. 256–273
[22] I. Mironov, O. Pandey, O. Reingold, S.P. Vadhan, Computational differential privacy, in *Advances in Cryptology—CRYPTO 2009*, ed By S. Halevi. *Lecture Notes in Computer Science*, vol. 5677 (Springer, 2009), pp. 126–142
[23] M.O. Rabin, How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981. Available online in the Cryptology ePrint Archive, Report 2005/187, arXiv:eprint.iacr.org/2005/187
[24] A.C. Yao, Protocols for secure computations, in *Proc. of the 23th IEEE Symp. on Foundations of Computer Science* (1982), pp. 160–164
[25] A.C. Yao, How to generate and exchange secrets, in *Proc. of the 27th IEEE Symp. on Foundations of Computer Science* (1986), pp. 162–167