Completeness for Symmetric Two-Party Functionalities - Revisited*

Yehuda Lindell[1], Eran Omri[2], and Hila Zarosim[1]

[1] Bar-Ilan University
lindell@biu.ac.il, zarosih@cs.biu.ac.il

[2] Ariel University Center
omrier@gmail.com

Abstract. Understanding the minimal assumptions required for carrying out cryptographic tasks is one of the fundamental goals of theoretical 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).
All previous works define reductions via an ideal implementation of the functionality; i.e., f reduces to g if one can implement f using an ideal 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 "ideal-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 n's). In light of the above, we propose an alternative definitional infrastructure for studying the triviality and completeness of functionalities.

1 Introduction

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, like *privacy* (meaning that nothing but the specified output is learned), *correctness* and more.

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

* 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. This work was done while Eran Omri was a postdoc at Bar-Ilan University.

of enhanced trapdoor permutations [17, 5]. Soon after, it was shown that any function can be securely computed, given an ideal box for computing the oblivious transfer function [7]. 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 [1, 9]. 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 [8, 11, 13]. For example, unlike the asymmetric setting, there exist (deterministic) symmetric functions that are neither complete nor trivial; see Figure 1 below for an example of such a function. This begs 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 [8] 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., [1, 7–9, 2, 6, 13]). 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.

All previous works define a reduction via an ideal implementation of a function; i.e., f_2 reduces to f_1 if a secure protocol for computing f_2 can be constructed given an ideal box (trusted party or oracle) that computes f_1 and gives the outputs to both parties simultaneously.[1] The advantage of (black-box) reductions of the above type is that they 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 necessary 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 reduction may be nonblack-box.

Our contributions. In this work we give substantial evidence that the picture of *computational hardness* of securely computing two-party functionalities in the presence of semi-honest adversaries is different to that drawn by the characterizations of completeness of [8, 11]. Specifically, we show that there exist symmetric functionalities f (i.e., where both parties get the same output), that are *not ideal-box-complete* (i.e., OT cannot be implemented using an ideal-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 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, an ideal-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 are functions that are neither complete nor trivial) and the asymmetric setting (where only one party receives output and all functions are either trivial or complete).

Alternative formulation of completeness – existential completeness. In light of the above, we propose a new definition of completeness that is not black box. We define the notion of an "achievable class" of a given functionality f. Informally speaking, the achievable class of a functionality f contains all

[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).

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.

Related Work. As we have already mentioned, completeness in secure two-party computation was investigated in a large body of work [2, 11, 8, 1, 10, 12, 9, 13, 15, 6]. We discuss a few that are more relevant to our discussion. Kilian [8] and Kushilevitz [11] consider the symmetric model and give criteria for the existence of unconditionally secure protocols [11] and for completeness [8]. Maji et al. [13] extended the discussion of the symmetric model to the UC-setting. Beimel, Malkin and Micali [1] considered the asymmetric model. They prove a zero-one law for completeness vs. triviality in this model. Almost all of these works consider functions with a constant size domain and information-theoretic security. The only exception is [6] who deals with computational security in the asymmetric model.

2 Definitions

2.1 Preliminaries

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 word.

Two ensembles $X = \{X(a, n)\}_{a \in \{0,1\}^*; n \in \mathbb{N}}$ and $Y = \{Y(a, n)\}_{a \in \{0,1\}^*; n \in \mathbb{N}}$ 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 $n \in \mathbb{N}$,

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

The ensembles X and Y are computationally indistinguishable by uniform machines, denoted $X \stackrel{c}{\equiv}_U Y$, if the above holds for every PPT distinguisher D.

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 [4]. 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 $P \in \{A, B\}$ 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 contains its private input, its random string, and the messages it received throughout this execution. The random variable $\mathrm{View}_A^\pi(x, y, 1^n)$ (respectively $\mathrm{View}_B^\pi(x, y, 1^n)$) 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 execution terminates. This pair is described by the random variable $\mathrm{Output}^\pi(x, y, 1^n) = (\mathrm{Output}_A^\pi(x, y, 1^n), \mathrm{Output}_B^\pi(x, y, 1^n))$, where $\mathrm{Output}_P^\pi(x, y, 1^n)$ 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 [4] for a motivating discussion regarding the definition.

Definition 2.1 (security for deterministic functionalities). *A protocol $\pi = \langle A, B\rangle$ securely computes a deterministic functionality $f = (f_A, f_B)$ 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*

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

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

Privacy: *There exist two probabilistic polynomial-time (in the security parameter) algorithms $\mathcal{S}_A, \mathcal{S}_B$ (called "simulators"), such that:*

$$\left\{\mathcal{S}_A(x, f_A(x, y), 1^n)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}} \stackrel{C}{\equiv} \left\{\mathrm{View}_A^\pi(x, y, 1^n)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}}, \tag{2}$$

$$\left\{\mathcal{S}_B(y, f_B(x, y), 1^n)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}} \stackrel{C}{\equiv} \left\{\mathrm{View}_B^\pi(x, y, 1^n)\right\}_{x,y\in\{0,1\}^*;n\in\mathbb{N}}. \tag{3}$$

For most of this paper, we will consider functionalities where both parties receive the same output, meaning that $f_A = f_B$. 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 from hereon.

Oblivious transfer – naive-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 [18, 5]. There are several equivalent

versions of OT; the most common being Rabin-OT [16] and 1-out-of-2 OT [3]. In this paper we use a slightly different version presented in [6], called Naive-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. We therefore need to define this weaker notion of security.

Definition 2.2 (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 Equations (1), (2) and (3) hold for* every *$n \in \mathcal{N}$, and Equations (2) and (3) hold with respect to uniform distinguishers.*

We stress that the correctness and privacy conditions must 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 preliminaries.

3.1 Preliminaries

Our theorem applies to all non-trivial functionalities, as characterized by Kushilevitz [11]. This characterization uses the notion of "decomposition" of a function. We now define this notion.

Definition 3.1 (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.*

Definition 3.2 (strongly non-decomposable functions). *A function* $g : X \times Y \to Z$ *is* strongly non-decomposable *if it is not monochromatic, all* $x \in X$ *are equivalent, and all* $y \in Y$ *are equivalent.*

We refer to [11] in order to see why this is called non-decomposable. The binary OR and AND functions are strongly non-decomposable, as is the function $f_{\mathcal{KUSH}}$ defined below:

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

Fig. 1. Kushilevitz's function $f_{\mathcal{KUSH}}$

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 3.3 (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.

We remark that Kushilevitz [11] proved that a function is non-trivial if and only if it is non-decomposable. The function $f_{\mathcal{KUSH}}$ is of particular interest since it is neither trivial (as shown by [11]) nor complete (as shown by [8]).

3.2 The Theorem and Proof

Let f be a symmetric non-decomposable functionality with domain of constant size. 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 special subset of inputs called a 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 2.2. We explain why this is the case at the end of Section 3.3.

Theorem 3.1. *If there exists a constant round protocol π that securely computes a symmetric, deterministic, non-decomposable functionality f (over a constant-size domain), then there exists a uniform infinitely-often* OT *protocol.*

Proof: 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 Section 3.3 we prove the first step. Specifically, in Lemma 3.1 we prove that there exists an "exclusive revelation round" which is a round in which one party has learned while the other has not, and then in Lemma 3.2 we prove that such a round must exist for inputs that form an insecure minor (defined below). We call this an "exclusive revelation minor". Next in Section 3.4 we prove that the existence of an exclusive revelation minor implies the existence of OT with weak correctness, and finally in Section 3.5 we explain how to boost the correctness and thus obtain full OT (with infinitely-often uniform security). □

3.3 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 $\text{View}_A^{\pi_\nu}(x, y, 1^n)$ and $\text{View}_B^{\pi_\nu}(x, y, 1^n)$ 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 3.4 (distinguishing between inputs). *Let π be a c-round protocol for computing a functionality f (where c is some function of the security parameter n), and fix $i \in \mathbb{N}$. For a triple x, y, y' of inputs, we say that $A(x)$* 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,*

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

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

As we will see below, it is crucial that D be a uniform PPT machine, since the parties need to run D in the OT protocol that we construct. For simplicity (and since it suffices for our needs), the above definition considers a fixed round i. This can be easily generalized to any (polynomial time computable) function $i : \mathbb{N} \to \mathbb{N}$ such that $i(n) \leq c(n)$ for every n.

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 3.4.

Definition 3.5 (exclusive-revelation round). *Let π be a 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 triplet x, y, y' such that $A(x)$ distinguishes between y and y' at round i, and $B(y)$ does not distinguish between x and x' at round i for any triplet x, x', y (we say that x, y, y' define the revelation round); or*
2. *There exists a triplet x, x', y such that $B(y)$ distinguishes between x and x' at round i and, $A(x)$ does not distinguish between y and y' at round i for any triplet x, y, y' (we say that x, x', y define the revelation round).*

Protocol π has an exclusive-revelation round *if there exists $0 \leq i \leq c$, 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 3.1. *Let f be a symmetric functionality that is not constant (and has domain of constant size). Let π be a constant-round protocol for securely computing f. Then, π has an exclusive-revelation round.*

Proof: For every (round number) $i \leq c$, every uniform PPT machine (distinguisher) D, and every triplet x, x', y (recall that there is a constant number of such triplets), we define

$$\varepsilon^{i,D}_{x,x',y}(n) = \left|\mathsf{Pr}\left[D\left(\mathrm{View}^{\pi_i}_B\left(x, y, 1^n\right), 1^n\right) = 1\right] - \mathsf{Pr}\left[D\left(\mathrm{View}^{\pi_i}_B\left(x', y, 1^n\right), 1^n\right) = 1\right]\right|$$

and let $r^D_{x,x',y}$ be the minimal round number $0 \leq i \leq c$ 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 $r^D_{x,x',y} = c+1$. Note that this means that $r^D_{x,x',y}$ is the first round such that the PPT machine D can distinguish the ensembles $\left\{\mathrm{View}^{\pi_i}_B\left(x, y, 1^n\right)\right\}_{n\in\mathbb{N}}$ and $\left\{\mathrm{View}^{\pi_i}_B\left(x', y, 1^n\right)\right\}_{n\in\mathbb{N}}$.

We further define $r_{x,x',y} = \min_D\left\{r^D_{x,x',y}\right\}$ (this is well defined, as every $r^D_{x,x',y} \in [c+1]$). 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 $B(y)$ distinguishes between x and x'). For every triplet x, y, y', we define $r_{x,y,y'}$ analogously.

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

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

Assume without loss of generality that $i^*_A \leq i^*_B$; we show that $i^*_A < i^*_B$. It suffices to show that $i^*_A < i^*_B$, since by the definition of i^*_B we know that $B(y)$ does not distinguish between x and x' at any round $i < i^*_B$ and for any triplet x, x', y. A crucial observation is that the view of a party does not change in the round that it is active, and hence, neither does its distinguishing capability. Hence, by the minimality of i^*_A, it must be that B is the one sending a message in round i^*_A, since otherwise A would be able to distinguish already in round $i^*_A - 1$. This means that B's view does not change in round i^*_A, and hence, by the minimality of i^*_B it cannot be that $i^*_A = i^*_B$. The case that $i^*_B \leq i^*_A$ is dealt with analogously. □

We complete this step of the proof 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. An insecure minor is a tuple of inputs x, x', y, y' such that $f(x,y) = f(x,y')$ and $f(x',y) \neq f(x',y')$ (X-minor), or $f(x,y) = f(x',y)$ and $f(x,y') \neq f(x',y')$ (Y-minor).

Definition 3.6 (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 3.2. *Let π be a protocol that securely computes a strongly non-decomposable symmetric function f with constant-size 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 constant-size domain. Assume that there exist x_j, y_k, y_ℓ (with $k < \ell$) that define an exclusive revelation at round i; that is, $A(x_j)$ 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 $y_{i_{t+1}} = y_\ell$. $A(x_j)$ distinguishes between y_{i_0} and $y_{i_{t+1}}$ at round i, and since t is a constant (recall that f has a constant-size domain), there exists some $h \in [t+1]$ such that $A(x_j)$ 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 infinitely many 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 3.1). Recall that an exclusive-revelation round is the first round that a party can distinguish between the inputs of the other party. 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 $\nu : \mathbb{N} \to \textit{round number}$, 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 the existence of it.

Constant-size domain. We restrict ourselves to functions with constant-size domains (i.e., not dependent on the security parameter) in order to be consistent with previous works studying completeness and triviality of symmetric functions ([8, 11]). Extending the study of completeness to functions with non-constant-size domains is beyond the scope of this paper.

3.4 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. Formally:

Definition 3.7. *A protocol π is a* infinitely-often uniform *weak oblivious transfer protocol (io-weak-OT) if there exists an infinite set $\mathcal{N} \subseteq \mathbb{N}$ such that Equations (2) and (3) hold for every $n \in \mathcal{N}$ and with respect to uniform distinguishers, and there exists a polynomial $p(\cdot)$ such that Equation (1) holds with probability $\frac{1}{2} + \frac{1}{p(n)}$ for every $n \in \mathcal{N}$.*

We stress that the privacy requirement of the oblivious transfer (Equations (2) and (3)) is identical to uniform infinitely-often security in Definition 2.2. However, 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 3.3. *Let $\pi = \langle A, B \rangle$ be a protocol for securely computing a functionality f. If π has an exclusive-revelation minor, then there exists a* PPT *protocol $\tilde{\pi}$ that is an infinitely-often uniform 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, such that x', y, y' define an exclusive revelation at round i for π (the case of an exclusive revelation Y-minor is analogous). That is, we have that $A(x')$ distinguishes between y and y' at round i and for *every* triplet $\hat{x}, \hat{x}', \hat{y}$, we have that $B(\hat{y})$ 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. Furthermore, since $f(x, y) = f(x, y')$ (by definition of a minor), by the security of π we also have that $A(x)$ 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

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

We now show how to construct an io-weak-OT protocol $\tilde{\pi}$. Before giving the formal description of the protocol, let us give some intuition. The idea is to run the protocol on the inputs of the above minor until round i, and then to halt the execution. By the exclusiveness of the revelation, we are guaranteed that B

learns nothing from the computation, hence the sender $\tilde{S}$ will play the role of B. If the receiver $\tilde{R}$ has input 0, then it will use x as its input and play the role of A, and hence will not learn anything (recall that $f(x,y) = f(x,y')$ and so the output reveals nothing about B's input, meaning that $\tilde{R}$ learns nothing). In case $\tilde{R}$'s input is 1 it will use x' as its input for the computation, and will learn the output by distinguishing as in Equation (4).

Regarding the sender's input, one possibility is to have the sender to use y' as its input for the computation in case $b = 0$ and y in case $b = 1$. The receiver will then output 0 or 1, depending on what 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 $1/2$). In order to overcome this, we have the sender use a random input in $\{y, y'\}$ and therefore transfer a random bit r to the receiver (which 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 1 (An io-weak-OT $\tilde{\pi} = \langle \tilde{S}, \tilde{R} \rangle$)

Inputs: *The private input of the sender $\tilde{S}$ is a bit $b \in \{0,1\}$ and the private input of the receiver $\tilde{R}$ 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 $\tilde{S}$ plays the role of B and the receiver $\tilde{R}$ plays the role of A. The inputs of the parties are set as follows:*
   - *The input of B (played by $\tilde{S}$) is y' if $r = 0$ and y if $r = 1$.*
   - *The input of A (played by $\tilde{R}$) is x if $c = 0$ and x' if $c = 1$.*

   The parties halt after the i-th round of π. Let v_A^i be the partial view of A in this partial execution of π.
3. *The sender $\tilde{S}$ sends $z = r \oplus b$ to the receiver $\tilde{R}$.*
4. *If $c = 0$, the receiver outputs λ. Otherwise (if $c = 1$), the receiver executes D on v_A^i, sets r' to be the output of D, and outputs $z \oplus r'$. The sender always outputs λ.*

Note that the receiver is allowed to use 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 $\mathsf{Pr}\left[\mathrm{Output}_{\tilde{R}}^{\tilde{\pi}}(b, c = 1, 1^n) = b\right] \geq$

$\frac{1}{2} + \frac{1}{q(n)}$. We will show that this holds for the polynomial $q(i) = 2p(i)$ and for all n's for which Equation (4) is satisfied. We fix such an n.

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

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

Since Equation (4) is satisfied for n, we have that

$$
\mathsf{Pr}\left[D\left(\mathrm{View}_A^{\pi_i}\left(x', y, 1^n\right), 1^n\right) = 1\right] - \mathsf{Pr}\left[D\left(\mathrm{View}_A^{\pi_i}\left(x', y', 1^n\right), 1^n\right) = 1\right] \geq \frac{1}{p(n)}.
$$

Hence, we conclude that $\mathsf{Pr}\left[r' = r\right] \geq \frac{1}{2} + \frac{1}{2p(n)}$, and so correctness holds.

Proving the *privacy* of the protocol. We now proceed to prove that Equations (2) and (3) in Definition 2.1 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 2.2). Due to the lack of space in this extended abstract, we sketch this portion of the proof.

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

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

The difference between the view of the sender in a real execution and in a simulation by $\mathcal{S}_{\tilde{S}}$ is due to the fact that $\mathcal{S}_{\tilde{S}}$ always runs A with x whereas in a real execution A runs with x or x' depending on the receiver's input. Nevertheless, these distributions are computationally indistinguishable since i is an *exclusive revelation round* for A. This means that B learns nothing about A's input up to and including round i, and in particular the view of B when A uses x is computationally indistinguishable from its view when A uses x'. We stress that the fact that i is an exclusive revelation round means that no *uniform distinguisher* given B's view can distinguish (by the notion of distinguishing between

inputs; Definition 3.4). This does not necessarily mean that no non-uniform distinguisher can distinguish; thus we only achieve privacy with respect to *uniform* distinguishers.

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 so 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 $\mathcal{S}_{\tilde{R}}$ receives as input the bit $c = 0$, the output $\text{OT}_R = \lambda$ of the functionality OT to the receiver, and the security parameter 1^n, and works as follows:

1. $\mathcal{S}_{\tilde{R}}$ executes π for i rounds, running A with input x and B with input y.
2. $\mathcal{S}_{\tilde{R}}$ chooses a random bit $z_{\mathcal{S}} \in \{0, 1\}$.
3. $\mathcal{S}_{\tilde{R}}$ outputs $z_{\mathcal{S}}$ appended to the partial view v_A^i 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. In order to see that this makes no difference, first observe that since x, x', y, y' form an insecure minor, it holds that $f(x, y) = f(x, y')$. Thus, when A has input x in an execution of π, it cannot distinguish the case that B used input y or y'; otherwise, A could learn something that is not revealed by the functionality output. Thus, the view of the receiver (who runs A) in the protocol execution is indistinguishable from its view in the simulation. Given the above, it follows that the distribution of a random bit $z_{\mathcal{S}}$ is indistinguishable from the distribution of $z = r \oplus b$ by the randomness of r. This completes 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 3.4, 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.

3.5 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 $\tilde{\pi}$ 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, $\tilde{\pi}$ is correct with probability $1-\mu(n)$, for some negligible function $\mu(\cdot)$. To prove the privacy of $\tilde{\pi}$, 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 3.1.

4 Ideal-Box and Existential 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. [8, 11, 1]), 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 all 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, who 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 [4, Section 7.3.1]. Formally, using the terminology of [4], all previous definitions say that g reduces to f if there exists an oracle-aided protocol π that *information-theoretically* securely computes g when using the oracle functionality f (the only exception is [6] that considers computational security rather than information-theoretic). A functionality f is called complete if all g reduce to it, and it is called trivial if it can be information-theoretically securely computed with no oracle. We call this notion ideal-box completeness since the reduction is *black-box in the functionality*.

The picture of completeness and triviality for the above definition is well known. Specifically, for the case of asymmetric functionalities where only one of the parties receives output, a functionality is complete if it contains an insecure minor, and trivial if not. Furthermore, for the case of symmetric functionalities where the parties receive the same output (i.e., $f_A = f_B$), a functionality is complete if and only if it contains an embedded OR, and is trivial if and only if it is decomposable (see Definition 3.3).

Combining the above with Theorem 3.1, we have the following corollary:

Corollary 4.1. *There exist symmetric deterministic functionalities over a domain of constant-size that are not neither trivial nor ideal-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 [7], one can show that any functionality can be securely computed with uniform infinitely-often security (Definition 2.2) 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 [**?**,**?**] hold also for infinitely-often OT).

Existential completeness – an alternative formulation. Corollary 4.1 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 ideal-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 4.1. *Let $\mathcal{U}$ denote the set of all polynomial-time computable functionalities. The* achievable class of $f \in \mathcal{U}$*, denoted as $\mathcal{C}(f)$, 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 $g \in \mathcal{C}(f)$. Functionality f is* existentially trivial *if $f \in \mathcal{C}(f_\lambda)$ (where $f_\lambda(\cdot,\cdot) = (\lambda, \lambda)$), and is* existentially complete *if $\mathcal{C}(f) = \mathcal{U}$.*

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 (ideal-box) computational completeness that appears in [6]. We also note that any functionality that is ideal-box-complete, or complete by the computation definition in [6], is also existentially complete.

We conclude by remarking that the definition of existential completeness 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 ideal-box definition (as hinted to by Corollary 4.1). 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 $g \in \mathcal{C}(f)$ does not necessarily tell us how to securely compute g, even given a protocol for securely computing f.

References

1. A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer-Verlag, 1999.
2. B. Chor and E. Kushilevitz. A zero-one law for Boolean privacy. *SIAM J. on Discrete Mathematics*, 4(1):36–47, 1991.
3. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.
4. O. Goldreich. *Foundations of Cryptography, Voume II – Basic Applications*. Cambridge University Press, 2004.
5. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Symp. on the Theory of Computing*, pages 218–229, 1987.
6. D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation: A computational view. In *Proc. of the 36th ACM Symp. on the Theory of Computing*, pages 252 – 261, 2004.
7. J. Kilian. Basing cryptography on oblivious transfer. In *Proc. of the 20th ACM Symp. on the Theory of Computing*, pages 20–31, 1988.
8. J. Kilian. A general completeness theorem for two-party games. In *Proc. of the 23th ACM Symp. on the Theory of Computing*, pages 553–560, 1991.
9. J. Kilian. More general completeness theorems for two-party games. In *Proc. of the 32nd ACM Symp. on the Theory of Computing*, pages 316–324, 2000.
10. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. on Computing*, 28(4):1189–1208, 2000. This is the journal version of [8, 12].
11. E. Kushilevitz. Privacy and communication complexity. *SIAM J. on Discrete Mathematics*, 5(2):273–284, 1992.
12. E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Proc. of the 35th IEEE Symp. on Foundations of Computer Science*, pages 478–491, 1994.
13. H.K. Maji, M. Prabhakaran, and M. Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. pages 256–273, 2009.
14. H.K. Maji, M. Prabhakaran, and M. Rosulek. A zero-one law for cryptographic complexity with respect to computational uc security. In *CRYPTO*, 2010.
15. I. Mironov, O. Pandey, O. Reingold, and S. P. Vadhan. Computational differential privacy. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142. Springer-Verlag, 2009.
16. 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, eprint.iacr.org/2005/187.
17. A. C. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 160–164, 1982.
18. A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.