Reducibility and Completeness In Private Computations*

Joe Kilian[†] Eyal Kushilevitz[‡] Silvio Micali[§] Rafail Ostrovsky[¶]

Abstract

We define the notions of *reducibility* and *completeness* in (two party and multi-party) private computations. Let g be an n-argument function. We say that a function f is *reducible* to a function g if n honest-but-curious players can compute the function f n-privately, given a black-box for g (for which they secretly give inputs and get the result of operating g on these inputs). We say that g is *complete* (for private computations) if *every* function f is reducible to g.

In this paper, we characterize the complete boolean functions: we show that a boolean function g is complete if and only if g itself cannot be computed n-privately (when there is no black-box available). Namely, for boolean functions, the notions of **completeness** and **n-privacy** are *complementary*. This characterization gives a huge collection of complete functions (*any* non-private boolean function!) compared to very few examples given (implicitly) in previous work. On the other hand, for non-boolean functions, we show that these two notions are *not* complementary.

1 Introduction

We consider (two party and multi-party) private computations. Quite informally, given an arbitrary n-argument function f, a t-private protocol should allow n players, each possessing an individual secret input, to satisfy simultaneously the following two constraints: (1) *(Correctness)*: all players learn the value of f and (2) *(Privacy)*: no set of at most t (faulty) players learns more about the initial inputs of other players than is implicitly revealed by f's output. This problem, also known as *secure computation*, have been examined in the literature with two substantially different types of faulty players – malicious (i.e. Byzantine) players and honest-but-curious players. Below we discuss some known results with respect to each of these two types of players.

Secure computation for malicious players. Malicious players may deviate from the prescribed protocol in an arbitrary manner, in order to violate the correctness and privacy constraints. The first

*This paper is based on (but not completely covers) two conference papers; a 1991 paper by Kilian [K-91] and a 1994 paper by Kushilevitz, Micali, and Ostrovsky [KMO-94].

[†] NEC Research Institute, New Jersey. E-mail: joe@research.nj.nec.com .

[‡] Department of Computer Science, Technion. Research supported by the E. and J. Bishop Research Fund and by the Fund for the Promotion of Research at the Technion. Part of this research was done while the author was at Aiken Computation Lab., Harvard University, Supported by research contracts ONR-N0001491-J-1981 and NSF-CCR-90-07677. E-mail: eyalk@cs.technion.ac.il .

[§] Laboratory for Computer Science, MIT. Supported by NSF Grant CCR-9121466.

[¶] Bell Communication Research, MCC-1C365B 445 South Street Morristown, New Jersey 07960-6438. E-mail: rafail@bellcore.com.

	honest-but-curious players	malicious players
computational model [Yao-82, GMW-87] (assuming trapdoor permutations exist)	$t \leq n$	$t < \frac{n}{2}$
private-channels model [BGW-88, CCD-88]	$t < \frac{n}{2}$	$t < \frac{n}{3}$

Figure 1: The number of faulty players, t, tolerable in each of the basic secure computation models (with n players)

general protocols for secure computation were given in [Yao-82, Yao-86] for the two-party case, and by [GMW-87] for the multi-party case. Other solutions were given in, e.g., [GHY-87, GV-87, BGW-88, CCD-88, BB-89, RB-89, CKOR-97] based on various assumptions (either intractability assumptions or physical assumptions such as the existence of private (untappable) communication channels between each pair of players). These solutions give t-privacy for $t < \frac{n}{2}$ or $t < \frac{n}{3}$ depending on the assumption made. (See Figure 1 for a summary of the main results.)

Secure computation for honest-but-curious players. Honest-but-curious players must always follow the protocol precisely but are allowed to "gossip" afterwards. Namely, some of the players may put together the information in their possession at the end of the execution in order to infer additional information about the original individual inputs. It should be realized that in this honest-but-curious model enforcing the *correctness* constraint is easy, but enforcing the *privacy* constraint is hard. The honest-but-curious scenario is not only interesting on its own (e.g., for modeling security against outside listeners or against passive adversary that wants to remain undetected); its importance also stems from "compiler-type" theorems, such as the one proved by [GMW-87][1] (with further extensions in many subsequent papers, for example, [BGW-88, CCD-88, RB-89]). This type of theorems provide algorithms for transforming any t-private protocol with respect to honest-but-curious players into a t'-private protocol with respect to malicious players ($t' \leq t$). Surprisingly, much of the research efforts were devoted to the more complicated case of malicious players, while the case of honest players is far from being well understood. In this paper we examine the latter setting.

Information theoretic privacy.[2] The information theoretic model was first examined by [BGW-88, CCD-88]. In particular, they prove that every function is $\lceil n/2 \rceil$-private (in the setting of honest-but-curious players; see Figure 1). The information theoretic model was then the subject of considerable work (e.g., [CKu-89, BB-89, CGK-90, CGK-92, CFGN-96, KOR-96, HM-97, BW-98]). Particularly, [CKu-89] characterized the boolean functions for which n-private protocols exist: an n-argument boolean function f is n-private if and only if it can be represented as $f(x_1, x_2, \ldots, x_n) = f_1(x_1) \oplus f_2(x_2) \oplus \ldots \oplus f_n(x_n)$ where each f_i is boolean. Namely, f is n-private if and only if it is the exclusive-or of n local functions. An immediate corollary of this is that *most* boolean functions are *not* n-private (even with respect to honest-but-curious players).

Our contribution. We formally define the notion of *reducibility* among multi-party protocol problems. We say that f is reducible to g, if there is a protocol that allows the n players to compute the value of f

[1] The reader is referred to [G-98] for a fully detailed treatment of the [Yao-82, GMW-87] results.

[2] in oppose to *computational-privacy*

n-privately, in the information theoretic sense, just by repeatedly using a black-box (or a trusted party) for computing g. That is, in any round of the protocol, the players *secretly* supply arguments to the black-box and then the black-box *publicly* announces the result of operating g on these arguments. We stress that the only means of communication among the players is by interacting with the black-box (i.e., evaluating g). For example, it is clear that *every* function is reducible to itself (all players secretly give their private inputs $x_1, \ldots, x_n$ to the black-box and it announces the result). Naturally, we can also define the notion of *completeness*. A function g is complete if *every* function f is reducible to g. The importance of this notion relies on the following observation:

> If g is complete, and g can be computed t-privately in some "reasonable" setting[3] (such as the settings of [GMW-87, BGW-88] etc.), then any function f can be computed t-privately in the *same* setting. Moreover, from our construction a stronger result follows: if in addition the implementation of g is *efficient* then so is the implementation of f (see below).

The above observation holds since our definition of reduction requires the highest level of privacy (n), the strongest notion of privacy (information theoretic), a simple use of g (black box), and it avoids making any (physical or computational) assumptions. Hence the straightforward simulation, in which each invocation of the black-box for g is replaced by an invocation of a "t-private" protocol for g, works in any "reasonable" setting (i.e. any setting which is not too weak to prevent simulation) and yields a "t-private" protocol for f. Previously, there was no easy way to translate protocols from one model (such as the models of [Yao-82, GMW-87, BGW-88, CCD-88, RB-89, FKN-94]) to other models.

It can be seen that if g is complete then g itself cannot be n-private. The inverse is the less obvious part: since the definition of completeness requires that the same function g will be used for computing *all* functions f, and since the definition of reductions seems very restrictive, it may be somewhat surprising that complete functions exist at all. Some examples of complete functions implicitly appear in the literature (without discussing the notions of reducibility and completeness). First such results were shown in [Yao-82, GMW-87, K-88].

In this work we prove the existence of complete functions for n-private computations. Moreover, while previous research concentrated on finding a *single* complete function, our main theorem *characterizes* all the *boolean* functions which are complete:

Main Theorem: For all $n \geq 2$, an n-argument boolean function g is complete if and only if g is *not* n-private.

Our result thus shows a very strong dichotomy: every boolean function g is either "simple enough" so that it can be computed n-privately (in the information theoretic model), or it is "sufficiently expressive" so that a black-box for it enables computing any function (not only boolean) n-privately (i.e., g is complete). We stress that there is no restriction on g, beside being non-n-private boolean function, and that no relation between the function g and the function f that we wish to compute is assumed. Note that using the characterization of [CKu-89] it is easy to determine whether a given boolean function g is complete. That is, a boolean function g is complete if and only if it cannot be represented as $g(x_1, x_2, \ldots, x_n) = g_1(x_1) \oplus g_2(x_2) \oplus \ldots \oplus g_n(x_n)$ where each g_i is boolean.

Some features of our result. To prove the completeness of a function g as above, we present an appropriate construction with the following additional properties:

- We consider the most interesting scenario, where both the reduced function, f, and the function g are n-argument functions (where n is the number of players). This enables us to organize the reduction in *rounds*, where in each round each player submits a value of a *single* argument to g (and the value of

[3] A *setting* consists of defining the type of communication, type of privacy, assumptions made etc.

each argument is supplied by exactly one player).[4] Thus, no player is "excluded" at any round from the evaluation of g. Our results however remain true even if the number of arguments of g is different from the number of arguments of f.

- Our construction evaluates the n-argument function g only on a *constant* number of n-tuples (hence, a partial implementation of g may be sufficient).
- When we talk about privacy, we put no computational restrictions on the power of the players; hence we get information-theoretic privacy. However, when we talk about protocols, we measure their *efficiency* in terms of the computational complexity of f (i.e., the size of a circuit that computes f); and in terms of a parameter k (our protocol allows error probability of $2^{-\Omega(k)}$). The protocol we introduce is efficient (polynomial) in all these measures.[5] We stress, though, that the n-tuples with which we use the function g are chosen non-uniformly (namely, they are encoded in the protocol) for the particular choices of g and n (the size of the network). These n-tuples do not depend though neither on the size of the *inputs* to the protocol nor on the function f.

Our main theorem gives a full characterization of the boolean functions g which are complete (those that are not n-private). When non-boolean functions are considered, it turns out that the above simple characterization is no longer true. That is, we show that there are (non-boolean) functions which are not n-private, yet are *not* complete.

Overview of the proof. Our proof goes along the following lines:

1. We define the notion of embedded-OR for two-argument functions and appropriately generalize this notion to the case of n-argument functions. We then show that if an n-argument function is not private then it contains an embedded-OR. For the case $n = 2$ this follows immediately from the characterization of [CKu-89]; the case $n > 2$ requires some additional technical work.
2. We show how an embedded-OR can be used to implement an *Oblivious Transfer* (OT) channel/primitive.[6] (It should be emphasized that an OT channel in a multi-party setting has the additional requirement that listeners do not get any information; we prove however that this property is already implied by the basic properties of two-party OT). Finally, it follows from the work of [GHY-87, GV-87, K-88, BG-89, GL-90] that n-private computation of any function f can be implemented given OT channels. All together, our main theorem follows.

Organization of the paper. In Section 2 we specify our model and provide some necessary definitions. In Section 3 we prove our main lemma that shows the existence of an embedded-OR in every non n-private, boolean function; In Section 4 we use the main lemma (i.e., the existence of an embedded-OR) to implement OT channels between players; In Section 5 we use the construction of OT channels to prove our main theorem. Finally, Section 6 contains a discussion of the results and some open problems. For completeness, we include in the appendix a known protocol for private computations using OT channels (including its formal proof).

[4] Which player submits which argument is a permutation specified by the reduction.

[5] Evaluating g on any assignment is assumed to take a unit time. All other operations (communication, computation steps, etc.) are measured in the regular way.

[6] Oblivious transfer is a protocol for two players: a sender that holds two bit b_0 and b_1 and a receiver that holds a selection bit s. At the end of the protocol the receiver gets the bit b_s but has no information about the value of the other bit, while the sender has no information about s.

2 Model and Definitions

Let f be an n-argument function defined over a finite domain $\mathcal{D}$. Consider a collection of $n \geq 2$ synchronous, computationally unbounded players $P_1, \ldots, P_n$ that communicate using a black-box for g, as described below. At the beginning of an execution, each player P_i has an input $x_i \in \mathcal{D}$. In addition, each player can flip unbiased and independent random coins. We denote by r_i the string of random bits flipped by P_i (sometimes we refer to the string r_i as the *random input* of P_i). The players wish to compute the value of a function $f(x_1, x_2, \ldots, x_n)$. To this end, they use a prescribed protocol $\mathcal{F}$. In the i-th round of the protocol, every processor P_j secretly sends a message m_j^i to the black-box g.[7] The protocol $\mathcal{F}$ specifies which argument to the black-box is provided by which player. The black-box then publicly announces the result of evaluating the function g on the input messages.

Formally, with each round i the protocol associates a permutation π_i. The value computed by the black-box at round i, denoted s_i, is $s_i = g(m^i_{\pi_i(1)}, m^i_{\pi_i(2)}, \ldots, m^i_{\pi_i(n)})$. Each message m_j^i, sent by P_j to the black-box in the i-th round, is determined by its input (x_j), its random input (r_j), and the output of the black-box in previous rounds ($s_1, \ldots, s_{i-1}$). We say that the protocol $\mathcal{F}$ computes the function f if the last value (or the last sequence of values in the case of non-boolean f) announced by the black-box equals the value of $f(x_1, x_2, \ldots, x_n)$ with probability $\geq 1 - 2^{-\Omega(k)}$, where k is a (confidence) parameter and the probability is over the choice of $r_1, \ldots, r_n$.

Let $\mathcal{F}$ be an n-party protocol, as described above. The *communication* $\mathtt{S}(\vec{x}, \vec{r})$ is the concatenation of all messages announced by the black-box, while executing $\mathcal{F}$ on inputs $x_1, \ldots, x_n$ and random inputs $r_1, \ldots, r_n$. We often consider the communication $\mathtt{S}$ while fixing $\vec{x}$ and some of the r_i's; in this case, the communication should be thought of as a random-variable where each of the r_i's that were not fixed is chosen according to the corresponding probability distribution. For example, if T is a set of players then $\mathtt{S}(\vec{x}, \{r_i\}_{i \in T})$ is a random variable describing the communication when each player P_i holds input x_i, each player in T holds random input r_i, and the random inputs for all players in $\overline{T}$ are chosen randomly. The definition of privacy considers the distribution of such random variables.

Definition 1 *Let $\mathcal{F}$ be an n-party protocol which computes a function f, and let $T \subseteq \{1, 2, \ldots, n\}$ be a set of players (coalition). We say that coalition T* does not learn any additional information *from the execution of $\mathcal{F}$ if the following holds: For every two input vectors $\vec{x}$ and $\vec{y}$ that agree on their T entries (i.e. $\forall i \in T : x_i = y_i$) and for which $f(\vec{x}) = f(\vec{y})$, for every choice of random inputs for the coalition's parties, $\{r_i\}_{i \in T}$, and for every communication S*

$$\Pr_{\{r_i\}_{i \in \overline{T}}}(\mathtt{S}(\vec{x}, \{r_i\}_{i \in T}) = S) = \Pr_{\{r_i\}_{i \in \overline{T}}}(\mathtt{S}(\vec{y}, \{r_i\}_{i \in T}) = S) \ .$$

Informally, this definition implies that for all inputs which "look the same" from the coalition's point of view (and for which, in particular, f has the same value), the communication also "look the same" (it is identically distributed). Therefore, by executing $\mathcal{F}$, the coalition T cannot infer any information on the inputs of $\overline{T}$, other than what follows from the inputs of T and the value of the function.

Definition 2 *A protocol $\mathcal{F}$ for computing f, using a black-box g, is* t-private *if any coalition T of at most t players does not learn any additional information from the execution of the protocol. A function f is* t-private *(with respect to the black-box g) if there exists a t-private protocol that uses the black-box g and computes f.*

Definition 3 *Let g be an n-argument function. We say that the black-box g (alternatively, the function g) is* complete *if every function f is n-private with respect to the black-box g.*

[7] Notice that we do not assume private point-to-point communication among players. On the other hand, we do allow private communication between players and the black-box for computing g.

Oblivious Transfer is a protocol for two players $\mathcal{S}$, the *sender*, and $\mathcal{R}$, the *receiver*. It was first defined by Rabin [R-81] and since then was studied in many works (e.g., [W-83, FMR-85, K-88, IL-89, OVY-91]). The variant of OT protocol that we use here, which is often referred to as $\binom{2}{1}$-OT, was originally defined in [EGL-85]. It was shown equivalent to other notions of OT (see, for example [R-81, EGL-85, BCR-86, B-86, C-87, K-88, CK-88]). The formalization of OT that we give is in terms of the probability distribution of the communication transcripts between the two players:

Definition 4 Oblivious Transfer (OT): *Let k be a (confidence) parameter. The sender $\mathcal{S}$ initially has two bits b_0 and b_1 and the receiver $\mathcal{R}$ has a selection bit c. After the protocol completion the following holds:*

Correctness: *Receiver $\mathcal{R}$ gets the value of b_c with probability greater than $1-2^{-\Omega(k)}$, where the probability is taken over the coin-tosses of $\mathcal{S}$ and $\mathcal{R}$. More formally, let $r_\mathcal{S}$, $r_\mathcal{R} \in \{0,1\}^{poly(k)}$ be the random tapes of $\mathcal{S}$ and $\mathcal{R}$ respectively, and denote the communication string by $\mathtt{comm}(\{b_0, b_1, c\}, \{r_\mathcal{S}, r_\mathcal{R}\}) \in \{0,1\}^{poly(k)}$. (Again, when one (or both) of $r_\mathcal{S}, r_\mathcal{R}$ is unspecified then $\mathtt{comm}$ becomes a random variable.) Then, for all k and for all $c, b_0, b_1 \in \{0,1\}$ the following holds:*

$$\Pr_{r_\mathcal{S}, r_\mathcal{R}}\left(\mathcal{R}(c, r_\mathcal{R}, \mathtt{comm}(\{b_0, b_1, c\}, \{r_\mathcal{S}, r_\mathcal{R}\})) = b_c\right) \quad \geq \quad 1 - \frac{1}{2^{\Omega(k)}}.$$

($\mathcal{R}(c, r_\mathcal{R}, comm)$ denotes the output of receiver $\mathcal{R}$ when it has a selection bit c, random input $r_\mathcal{R}$ and the communication in the protocol is comm.)

Sender's Privacy: *Receiver $\mathcal{R}$ does not get any information about b_{1-c}. (In other words, $\mathcal{R}$ has the "same view" in the case where $b_{1-c} = 0$ and in the case where $b_{1-c} = 1$). Formally, for all k, for all $c, b_c \in \{0,1\}$, for all $r_\mathcal{R}$ and for all communication comm:*

$$\Pr_{r_\mathcal{S}}\left(\mathtt{comm}(\{b_c, b_{1-c} = 0, c\}, r_\mathcal{R}) = comm\right) \quad = \quad \Pr_{r_\mathcal{S}}\left(\mathtt{comm}(\{b_c, b_{1-c} = 1, c\}, r_\mathcal{R}) = comm\right).$$

Receiver's Privacy: *Sender $\mathcal{S}$ does not get any information about c. (In other words, $\mathcal{S}$ has the "same view" in the case where $c = 0$ and in the case where $c = 1$). Formally, for all k, for all $b_0, b_1 \in \{0,1\}$, for all $r_\mathcal{S}$ and for all communication comm:*

$$\Pr_{r_\mathcal{R}}\left(\mathtt{comm}(\{b_0, b_1, c = 0\}, r_\mathcal{S}) = comm\right) \quad = \quad \Pr_{r_\mathcal{R}}\left(\mathtt{comm}(\{b_0, b_1, c = 1\}, r_\mathcal{S}) = comm\right).$$

REMARK: We emphasize that both $\mathcal{S}$ and $\mathcal{R}$ are *honest* (but curious) and assumed to follow the protocol. When OT is defined with respect to *cheating* players, it is usually allowed that with probability $2^{-\Omega(k)}$ information will leak. This however is not needed for honest players.

3 A New Characterization of n-private Boolean Functions

In this section we prove our main lemma which establishes a new combinatorial characterization of the family of n-private boolean functions. First, we define what it means for a two-argument boolean function to have an "embedded-OR" and use [CKu-89] to claim that any two-argument boolean function which is not 1-private contains an embedded-OR. We then generalize the definition and the claim to multi-argument functions in the appropriate way.

Definition 5 *We say that a two-argument function h contains an embedded-OR if there exist inputs x_0, x_1, y_0, y_1 ($x_0 \neq x_1$, $y_0 \neq y_1$) and an output value σ such that $h(x_1, y_1) = h(x_1, y_0) = h(x_0, y_1) = \sigma$ but $h(x_0, y_0) \neq \sigma$.*

Definition 6 *We say that an n-argument ($n \geq 3$) function f contains an embedded-OR if there exist indices $1 \leq i < j \leq n$, and values a_k for all $k \notin \{i, j\}$, such that the two-argument function*

$$h(y, z) \stackrel{\triangle}{=} f(a_1, \ldots, a_{i-1}, y, a_{i+1}, \ldots, a_{j-1}, z, a_{j+1}, \ldots, a_n)$$

contains an embedded-OR.

The following facts are proven in [CKu-89] (or follow trivially from it):

1. An n-argument boolean function is $\lceil n/2 \rceil$-private if and only if it can be written as $f(x_1, \ldots, x_n) = f_1(x_1) \oplus \ldots \oplus f_n(x_n)$, where each f_i is boolean.
2. A two-argument boolean function f is not 1-private if and only if it contains an embedded-OR.
3. If an n-argument boolean function is $\lceil n/2 \rceil$-private then it is n-private.
4. An n-argument boolean function f is $\lceil n/2 \rceil$-private if and only if in every partition of the indices $\{1, \ldots, n\}$ into two sets $S, \bar{S}$, each of size at most $\lceil n/2 \rceil$, the two-argument boolean function f_S defined by
$$f_S(\{x_i\}_{i \in S}, \{x_i\}_{i \in \bar{S}}) \stackrel{\triangle}{=} f(x_1, \ldots, x_n) \ ,$$
is 1-private.

Our main lemma extends Fact 2 above to the case of multi-argument functions.

Lemma 1 (Main Lemma:) *Let $g(x_1 \ldots, x_n)$ be any boolean, n-argument function. The function g is not $\lceil n/2 \rceil$-private if and only if it contains an embedded-OR.*

Proof: Clearly, if g contains an embedded-OR then there is a partition of the indices, as in Fact 4, such that the corresponding two-argument function g_S contains an embedded-OR (e.g., if i, j are the indices guaranteed by Definition 6 then include the index i in S, the index j in $\bar{S}$, and partition the other $n-2$ indices arbitrarily into two halves between S and $\bar{S}$). Hence, g_S is not 1-private and so, by Fact 4, g is not $\lceil n/2 \rceil$-private.

For the other direction, since g is not $\lceil n/2 \rceil$-private then, again by Fact 4, there is a partition $S, \bar{S}$ of the indices $\{1, \ldots, n\}$ such that g_S is not 1-private. For simplicity of notations, we assume that n is even and that $S = \{1, \ldots, n/2\}$. By Fact 2, the two-argument function g_S contains an embedded-OR. Hence, by Definition 5, there exist inputs u, v, w, z and a value $\sigma \in \{0, 1\}$ which form the following structure:

$g_S(\cdot, \cdot)$	$w = w_{\frac{n}{2}+1}, \ldots, w_n$	$z = z_{\frac{n}{2}+1}, \ldots, z_n$
$u = u_1, \ldots, u_{\frac{n}{2}}$	σ	σ
$v = v_1, \ldots, v_{\frac{n}{2}}$	σ	$\bar{\sigma}$

where $u \neq v$ and $w \neq z$. To complete the proof, we will show below that it is possible to choose these four inputs so that $u_i \neq v_i$ for exactly one coordinate i and $w_j \neq z_j$ for exactly one coordinate j (this will show that g satisfies the condition of Definition 6). To this end, we will first show how based on the inputs above we can find u' and v' which are different in exactly one coordinate. Then, based on the new u', v' and a similar argument, we can find w', z' which are different in exactly one coordinate. All this process is done in a way that maintains the OR-like structure, and therefore, by using the above values of i, j, fixing all the other arguments in S to $u'_k = v'_k$ and all the other arguments in $\bar{S}$ to $w'_k = z'_k$, we get that g itself contains an embedded-OR.

Let $L \subseteq \{1, \ldots, \frac{n}{2}\}$ be the set of indices on which u and v disagree (i.e., indices k such that $u_k \neq v_k$). Define the following sets of vectors: T_m is the set of all vectors that can be obtained from the vector u by replacing the value u_k in exactly m coordinates from L (in which $v_k \neq u_k$) by the value v_k. In particular, $T_0 = \{u\}$ and $T_{|L|} = \{v\}$. In addition, we define the following two sets of vectors:

$$X_1 \stackrel{\triangle}{=} \{x = (x_1, ..., x_{n/2}) \mid g_S(x, w) = g_S(x, z)\}$$

and

$$X_2 \stackrel{\triangle}{=} \{x = (x_1, ..., x_{n/2}) \mid g_S(x, w) \neq g_S(x, z)\},$$

where w and z are the specific vectors we choose above. In particular, we have $u \in X_1$ and $v \in X_2$.

We now claim that there must exist u', v' as required. Namely, the vector u' is in X_1, the vector v' is in X_2 and u', v' differ in exactly one coordinate. Suppose, towards a contradiction, that this is not true (i.e., no such u', v' exist). We will show that this implies that $T_m \subseteq X_1$, for all $0 \leq m \leq |L|$, contradicting the fact that v which is in $T_{|L|}$ belongs to X_2. The proof is by induction. It is true for $m = 0$ as T_0 contains only u which is in X_1. Suppose the induction hypothesis holds for m. That is, $T_m \subseteq X_1$. For each vector x in T_{m+1}, there is a vector in T_m which differs from x in exactly one coordinate. Since we assumed that u', v' as above do not exist, this immediately implies that x is also in X_1 hence $T_{m+1} \subseteq X_1$, as needed. Therefore, we reached a contradiction which implies the existence of u', v' as required. That is, we found u', v' that differ in a single index i (i.e., $u'_i \neq v'_i$) and such that u', v', w, z still form an OR-like structure:

$g_S(\cdot, \cdot)$	$w = w_{\frac{n}{2}+1}, \ldots, w_n$	$z = z_{\frac{n}{2}+1}, \ldots, z_n$
$u' = u'_1, \ldots, u'_{i-1}, u'_i, u'_{i+1}, \ldots, u'_{\frac{n}{2}}$	σ	σ
$v' = u'_1, \ldots, u'_{i-1}, v'_i, u'_{i+1}, \ldots, u'_{\frac{n}{2}}$	σ	$\bar{\sigma}$

A similar argument shows the existence of w', z' that differ in a single index j and such that the vectors u', v', w' and z' form an OR-like structure:

$g_S(\cdot, \cdot)$	$w' = w'_{\frac{n}{2}+1}, \ldots, w'_{j-1}, w'_j, w'_{j+1}, \ldots, w'_n$	$z = w'_{\frac{n}{2}+1}, \ldots, w'_{j-1}, z'_j, w'_{j+1}, \ldots, w'_n$
$u' = u'_1, \ldots, u'_{i-1}, u'_i, u'_{i+1}, \ldots, u'_{\frac{n}{2}}$	σ	σ
$v' = u'_1, \ldots, u'_{i-1}, v'_i, u'_{i+1}, \ldots, u'_{\frac{n}{2}}$	σ	$\bar{\sigma}$

This shows that g contains an embedded-OR (with indices i, j as required by Definition 6). □

4 Constructing Embedded Oblivious Transfer

The first, very simple, observation is that given a black-box for a function g that contains an embedded-OR, we can actually compute the OR of two bits. That is, suppose that the n players wish to compute $\text{OR}(b_k, b_\ell)$ where b_k is a bit held by player P_k and b_ℓ is a bit held by player P_ℓ. Let i, j, x_0, x_1, y_0, y_1 be the indices and inputs as guaranteed by Definitions 5 and 6. Then, player P_k will provide the black box with the i-th argument which is x_{b_k} (i.e., if $b_k = 0$ then the argument provided by P_k is x_0 and if $b_k = 1$ then the argument is x_1) and player P_ℓ will provide the black box with the j-th argument which is x_{b_ℓ}. The other $n - 2$ players will provide the $n - 2$ fixed values $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n$ in an arbitrary order. The black-box will answer with the value

$$g(a_1, \ldots, a_{i-1}, x_{b_k}, a_{i+1}, \ldots, a_{j-1}, x_{b_\ell}, a_{j+1}, \ldots, a_n)$$

which is σ if $\mathrm{OR}(b_k, b_\ell) = 1$ and is different than σ if $\mathrm{OR}(b_k, b_\ell) = 0$. Hence, we showed how to compute $\mathrm{OR}(b_k, b_\ell)$.

Our main goal in this section is to show how, based on a black-box that can compute OR we can implement an Oblivious Transfer (OT) protocol. We start with the two-party case ($n = 2$) and then proceed to the general case which builds upon the two-party case.

4.1 The Two-Party Case

In this section we show how to implement a two-party OT protocol. We start by implementing a variant of OT, called *random OT* (or ROT for short), which is different than the standard OT (i.e., $\binom{2}{1}$-OT). In a ROT protocol the sender $\mathcal{S}$ has a bit s to be sent. At the end of the protocol, the receiver $\mathcal{R}$ gets a bit s' such that with probability $1/2$ the bit s' equals s and with probability $1/2$ the bit s' is random. The receiver knows which of the two cases happened and the sender has no idea which is the case. We start with a formal definition of the ROT primitive:

Definition 7 Random Oblivious Transfer (ROT): *Let k be a (confidence) parameter. The sender $\mathcal{S}$ initially has a single input bit s (and the receiver has no input). After the protocol completion the following holds:*

Correctness: *With probability greater than $1 - 2^{-\Omega(k)}$, receiver $\mathcal{R}$ outputs a pair of bits (I, s'), where I is referred to as the* indicator *(otherwise $\mathcal{R}$ outputs* `fail`*). (As usual, the probability is taken over the coin-tosses of $\mathcal{S}$ and $\mathcal{R}$, i.e., $r_\mathcal{S}$, $r_\mathcal{R} \in \{0,1\}^{poly(k)}$.) Moreover, if the output of $\mathcal{R}$ satisfies $\mathcal{R}(r_\mathcal{R}, \mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R})) = (1, s')$ (i.e., $I = 1$) then $s' = s$, while if $\mathcal{R}(r_\mathcal{R}, \mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R})) = (0, s')$ (i.e., $I = 0$) then s' is random; that is,*

$$\mathrm{Pr}_{r_\mathcal{S}, r_\mathcal{R}} \left(s' = 1 | \mathcal{R}(r_\mathcal{R}, \mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R})) = (0, s') \right) \quad = \quad \frac{1}{2}.$$

Sender's Privacy: *The probability that $\mathcal{R}$ outputs a pair (I, s') such that $I = 1$ is exactly $1/2$. That is,*

$$\mathrm{Pr}_{r_\mathcal{S}, r_\mathcal{R}} \left(\mathcal{R}(r_\mathcal{R}, \mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R})) = (1, s') \right) \quad = \quad \mathrm{Pr}_{r_\mathcal{S}, r_\mathcal{R}} \left(\mathcal{R}(r_\mathcal{R}, \mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R})) = (0, s') \right) \quad = \quad \frac{1}{2}.$$

Receiver's Privacy: *Sender $\mathcal{S}$ does not get any information about I. (In other words, $\mathcal{S}$ has the "same view" in the case where $I = 0$ and in the case where $I = 1$). Formally, for all k, for all $s \in \{0,1\}$, for all $r_\mathcal{S}$ and for all communication comm:*

$$\mathrm{Pr}_{r_\mathcal{R}} \left(\mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R}) = comm \mid I = 0 \right) \quad = \quad \mathrm{Pr}_{r_\mathcal{R}} \left(\mathtt{comm}(s, r_\mathcal{S}, r_\mathcal{R}) = comm \mid I = 1 \right).$$

Transformations of ROT protocols to $\binom{2}{1}$-OT protocols are well-known [C-87].[8] Our ROT protocol is implemented as follows:

[8] Assume that the sender, $\mathcal{S}$, has two bits b_0, b_1 and the receiver, $\mathcal{R}$, has a selection bit c. The players $\mathcal{S}$ and $\mathcal{R}$ repeat the following for at most $m = \Theta(k)$ times: at each time $\mathcal{S}$ tries to send to $\mathcal{R}$ a pair of random bits (s_1, s_2) using two invocations of ROT. If in both trials the receiver gets the actual bit or in both trials he gets a random bit then they try for another time. If the receiver got exactly one of s_1 and s_2 he sends the sender a permutation of the indices π (i.e., either $(1,2)$ or $(2,1)$) such that $s_{\pi(c)}$ is known to him. The sender replies with $b_1 \oplus s_{\pi(1)}, b_2 \oplus s_{\pi(2)}$. The receiver can now retrieve the bit b_c and knows nothing about the other bit. The sender, by observing π learns nothing about c (since he does not know from the invocation of the ROT protocols in which invocation the receiver got the actual bit and in which he got a random bit). Thus, we get a $\binom{2}{1}$-OT protocol based on the ROT protocol.

a. The sender, $\mathcal{S}$, and the receiver, $\mathcal{R}$, repeat the following until $c_1 = c_2 = 1$ (and at most $m = \Theta(k)$ times):
$\mathcal{S}$ chooses a pair (a_1, a_2) out of the two pairs $\{(1,0),(0,1)\}$, each with probability 1/2.
$\mathcal{R}$ chooses a pair (b_1, b_2) out of the three pairs $\{(1,0),(0,1),(1,1)\}$, each with probability 1/3.
$\mathcal{S}$ and $\mathcal{R}$ compute (using the black-box) $c_1 = \mathrm{OR}(a_1, b_1)$ and $c_2 = \mathrm{OR}(a_2, b_2)$.

b. If $c_1 = c_2 = 1$ then $\mathcal{S}$ sends $w = s \oplus a_1$ to $\mathcal{R}$. The receiver $\mathcal{R}$ outputs $I = 0$ if $(b_1, b_2) = (1,1)$ and outputs $I = 1$ otherwise; in addition, $\mathcal{R}$ outputs $s' = w \oplus b_2$.

c. If in all m times no choices (a_1, a_2) and (b_1, b_2) are such that $c_1 = c_2 = 1$ the protocol halts and $\mathcal{R}$ outputs `fail`.

To analyze the protocol we observe the following properties of it:

1. If $(b_1, b_2) = (a_1, a_2)$ then one of c_1, c_2 is 0. This happens in two of the six choices of (a_1, a_2) and (b_1, b_2), i.e., with probability 1/3. In each of the other four choices we get $c_1 = c_2 = 1$. Therefore, the probability of failure in $m = \Theta(k)$ trials is exponentially small.

2. Conditioned on the case $c_1 = c_2 = 1$, we have $(b_1, b_2) = (a_2, a_1)$ with probability 1/2 (two out of the four remaining cases) and $(b_1, b_2) = (1,1)$ with probability 1/2.

3. In case that $(b_1, b_2) = (a_2, a_1)$, we have in particular $b_2 = a_1$ and so $s' = w \oplus b_2 = (s \oplus a_1) \oplus b_2 = s$. In this case $\mathcal{R}$ outputs $I = 1$, as needed.
In case that $(b_1, b_2) = (1,1)$, each of the two choices of (a_1, a_2) is equally likely and therefore a_1 and hence also w and s' are random (i.e., each has the value 0 with probability 1/2 and the value 1 with probability 1/2). In this case $\mathcal{R}$ outputs $I = 0$, as needed.

4. As argued in 3, if the protocol does not fail then $\mathcal{R}$ knows the "correct" value of I (since he knows the values of b_1, b_2, c_1 and c_2). The sender, on the other hand, based on (a_1, a_2) cannot know which of the two equally-probable events, $(b_1, b_2) = (a_2, a_1)$ or $(b_1, b_2) = (1,1)$, happened and therefore he sees the same view whether we are in the case $I = 1$ or in the case $I = 0$.

Properties 1 and 3 above imply the correctness of the ROT protocol while properties 2 and 4 imply the sender's privacy and receiver's privacy (respectively). Hence, combining the above construction (including the transformation of the ROT protocol to a $\binom{2}{1}$-OT protocol) with Lemma 1, we get:

Lemma 2 *An OT-channel between two players is realizable given a black-box g, for any non-2-private function g.*

4.2 The Multi-Party Case ($n > 2$)

We have shown in our main lemma (Lemma 1) that any non n-private function g contains an embedded-OR. Thus, as explained above, we can use the black-box for g to compute the OR of two bits held by two players P_k and P_ℓ (where the other $n-2$ players assist by specifying the fixed arguments given by our main lemma). Then, based on the ability to compute OR, we showed in Section 4.1 above how any two players can implement an OT channel between them in a way that satisfies the properties of OT (in particular, the privacy of the sender and the receiver with respect to each other). However, there is a subtle difficulty in implementing a private OT-channel in a multi-player system which we must address: beside the usual properties of an OT channel (as specified by Definition 4), we should guarantee that the information transmitted between the two owners of the channel will not be revealed to potential *listeners* (i.e., the other $n-2$ players). If the OT channel is implemented "physically" then clearly no information is revealed to the listeners. However, since we implement OT using a black-box to some function g, which *publicly* announces each of its outcomes, we must also prove that this reveals no information to the listeners. That is, the communication comm should be distributed in the same way, for all values of b_1, b_2 and c.

The following lemma shows that the security of the OT protocol with respect to listeners is, in fact, already guaranteed by the basic properties of the OT protocol; namely, the security of the protocol with respect to both the receiver and the sender.

Lemma 3 *Consider any (two-player) OT protocol. For every possible communication comm, the probability* $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0,b_1,c\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$ *is the same for all values* b_0 *and* b_1 *for the sender and* c *for the receiver. (In other words, a listener sees the same probability distribution of communications no matter what are the inputs held by the sender and the receiver in the OT protocol.)*

Proof: Consider the following 8 probabilities corresponding to all possible values of b_0, b_1 and c:

1. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=0,b_1=0,c=0\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
2. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=0,b_1=0,c=1\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
3. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=0,b_1=1,c=0\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
4. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=0,b_1=1,c=1\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
5. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=1,b_1=0,c=0\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
6. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=1,b_1=0,c=1\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
7. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=1,b_1=1,c=0\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$
8. $\Pr_{r_{\mathcal{S}},r_{\mathcal{R}}}(\texttt{comm}(\{b_0=1,b_1=1,c=1\},\{r_{\mathcal{S}},r_{\mathcal{R}}\}) = comm)$

The receiver's privacy property implies that the terms (1) and (2) are equal, (3) and (4) are equal, (5) and (6) are equal, and (7) and (8) are equal. The sender's privacy property implies that the terms (1) and (3) are equal, (5) and (7) are equal, (2) and (6) are equal, and (4) and (8) are equal. All together, we get that all 8 probabilities are equal, as desired. □

5 A Completeness Theorem for Multi-Party Boolean Black-Box Reductions

In this section we state the main theorem and provide its proof. It is based on a protocol that can tolerate $n-1$ honest-but-curious players, assuming the existence of an OT-channel between each pair of players. Such protocols appear in [GHY-87, GV-87, K-88, BG-89, GL-90] (these works deal also with malicious players). That is, by these works we get the following lemma (for self-containment, both a protocol and its proof of security appear in the appendix):

Lemma 4 *Given OT channels between each pair of players, any* n*-argument function* f *can be computed* n*-privately (in time polynomial in the size of a boolean circuit for* f*).*

We are now ready to state our main theorem:

Theorem 1 (MAIN:) *Let $n \geq 2$ and let g be an n-argument boolean function. The function g is complete if and only if it is* not *n-private.*

Proof:
($\Longrightarrow$) First, we show that any complete g cannot be n-private. Towards the contradiction let us assume that there exists such a function g which is n-private and complete. This implies that all functions are n-private (as instead of using the black-box g the players can evaluate g by using the n-private protocol for g). This however contradicts the results of [BGW-88, CKu-89] that show the existence of functions which are *not* n-private.

($\Longleftarrow$) Next (and this is where the bulk of the work is) we show how to compute any function n-privately, given a black-box for any g which is not n-private. Recall that there exists a protocol that can tolerate $n-1$ honest-but-curious players, assuming the existence of OT-channels (Lemma 4). Also, we have shown how a black-box, computing *any* non-private function, can be used to simulate OT channels (Lemma 2 and 3). Combining all together we get the result. □

The theorem implies that "most" boolean functions are complete. That is, any boolean function which is not of the XOR-form of [CKu-89] is complete.

6 Conclusions and Further Extensions

6.1 Non-boolean Functions

We have shown that *any* non-n-private boolean function g is complete. Namely, a black-box for such a function g can be used for computing any function f n-privately. Finally, let us briefly turn our attention to non-boolean functions. First, we emphasize that if a function g contains an embedded-OR then it is still complete even if it is non-boolean (all the arguments go through as they are; in particular note that Definitions 5 and 6 of embedded-OR apply for the non-boolean case as well). For the non-boolean case, we can state the following proposition:

Proposition 2 *For every $n \geq 2$ there exists a (non-boolean) n-argument function g which is not n-private, yet such that g is* not *complete.*

Proof: The proof for 2-argument g is as follows: there are non-private two-argument functions which do not contain an embedded OR. Examples of such functions were shown in [Ku-89] (see Figure 2). We now show that with no embedded-OR one cannot compute the OR function. Assume, towards a contradiction, that there is some two-argument function f which does not have an embedded-OR, yet it could be used to compute the OR function. Since f can be used to compute the OR function, we can use it to implement OT (Lemma 2). Hence, there exists an implementation of OT based on some f which does not have an embedded-OR. However, [K-91] has shown that for two-argument functions, only the ones that contain an embedded-OR, can be used to implement OT, deriving a contradiction.

For n-argument functions, notice that if we define a function g (on n arguments) to depend only on its first two arguments, we are back to the 2-argument case, as the resulting function is not n-private. □

To conclude, we have shown that for boolean case, the notions of **completeness** and **privacy** are *exactly complementary*, while for the non-boolean case they are *not*.

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

Figure 2: A non-private function which does not contain an embedded-OR

6.2 Additional Remarks

In this section, we briefly discuss some possible extensions and easy generalizations of our results.

The first issue that we address is the need for the protocol to specify the permutation π_i that is used in each round i (for mapping the players to the arguments for the black-box g). Note that in our construction, we use the black-box only for computing the OR function on two arguments. For this, we need to map some two players P_k and P_ℓ, holding these two arguments, to the special coordinates i, j, guaranteed by the definition of embedded-OR. Therefore, without loss of generality, the sequence of permutations can be made oblivious (i.e., independent of the function f computed) at a price of $O(n^2)$ multiplicative factor to the rounds (and time). Moreover, at a price of $O(n^4)$ the sequence of permutations can even be made *independent* of the non-n-private function g. Finally, note that if g is a symmetric function (which is often the "interesting" case), then there is no need to permute the inputs to g.

Next, we recall the assumption that the number of arguments of g is the same as the number of arguments of f (i.e., n). Again, it follows from our constructions that this is not essential to any of our results: all that is needed is the ability for the two players P_k, P_ℓ, that wish to compute the OR function in a certain step, to do so by providing the two distinguished arguments i, j for g and all the other (fixed) arguments can be provided by arbitrary players (e.g., all of them by P_1).

In our definitions we require *perfect* privacy. That is, we require that the two distributions in Definition 1 are identical. One can relax this definition of privacy to require only *statistical indistinguishability* of distributions or only *computational indistinguishability* of distributions. For these definitions we refer the reader to the papers mentioned in the introduction. Note that if f can be computed "privately", under any of these notions, using a black-box for g and if g can be computed t-privately, under any of these notions, then also the function f can be computed t-privately, under the appropriate notion of privacy (i.e., the weaker among the two).

Finally, we note that the negative result of [CKu-89] allows a probability of error; hence, even a weaker notion of reduction that allows for errors in computing f does not change the family of complete functions. This impossibility result (i.e., first direction of the main theorem) still holds even if we allow the players to communicate not only using the black-box but also using other types of communication such as point-to-point communication channels.

6.3 Open Questions

The above results can be easily extended to show that any boolean g which is complete can also be used for a private computation of any *multi-output* function f (i.e., a function whose output is an n-tuple $(y_1, \ldots, y_n)$, where y_i is the output that should be given to P_i). This is so, because Lemma 4 still holds. On the other hand, it is an interesting question to characterize the *multi-output* functions g that are complete (even in the boolean case where each output of g is in $\{0, 1\}$).

It is not clear how to extend the model and the results to the case of *malicious* players in its full generality. Notice, however, that under the appropriate definition of the model, if we are given as a black-box the two-argument OR function we can still implement private channels (see [KMO-94] for details), and hence by [BGW-88, CCD-88] can implement any f, $n/3$-privately with respect to malicious players.

Suppose that we relax the notion of privacy to computational-privacy (as in [Yao-82, GMW-87]). In such a case, any computationally n-private implementation of an (information-theoretically) non-n-private (equivalently, complete) boolean function g implies the existence of a one-way function. This is so, since we have shown that such an implementation of g implies an implementation of OT, which in turn implies the existence of a one-way function by [IL-89]. However, the best known implementation of such protocols, for a function g as above, requires *trapdoor* one-way permutations [GMW-87]. It is an important question whether there exists an implementation based on a one-way function (or permutation) for functions without trap-door. This question has only some partial answers. In particular, when one of the players has super-polynomial power, this is possible [OVY-91]. However, if we focus on polynomial-time players and protocols, then the result of our paper together with the work of [IR-89] implies that for all *complete* functions, if we use only black-box reductions, this is as difficult as separating $\mathcal{P}$ from $\mathcal{NP}$. Thus, using black-box reductions, *complete* functions seem to be hard to implement (with computational privacy) without a trapdoor property. Notice, however, that for non-boolean functions we have shown that there are functions which are not n-private and not complete. It is not known even if these functions can be implemented without using trapdoor, although the results of [IR-89] do not apply to this case.

ACKNOWLEDGMENTS: We wish to thank Oded Goldreich for helpful discussions and very useful comments. We thank Mihir Bellare for pointing out to us in 1991 that the works of Chor, Kushilevitz and Kilian are complementary and thus imply a special case of our general result. Finally, we thank Amos Beimel for helpful comments.

References

[BB-89] J. Bar-Ilan, and D. Beaver, *Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds*, Proc. of 8th PODC, 1989, pp. 201-209.

[BGW-88] M. Ben-or, S. Goldwasser, and A. Wigderson, *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. of 20th STOC, 1988, pp. 1-10.

[B-86] M. Blum, *Applications of Oblivious Transfer*, Unpublished manuscript.

[BCC-88] G. Brassard, D. Chaum and C. Crépeau, *Minimum Disclosure Proofs of Knowledge*, JCSS, v. 37, pp 156-189.

[BCR-86] G. Brassard, C. Crépeau and J.-M. Robert, *Information Theoretic Reductions among Disclosure Problems*, IEEE Symp. on Foundations of Computer Science, 1986 pp. 168-173.

[BG-89] D. Beaver, and S. Goldwasser, *Multiparty Computation with Faulty Majority*, FOCS 1989.

[BW-98] D. Beaver, and A. Wool, *Quorum-based Secure Multi-Party Computation*, EuroCrypt 1998.

[CFGN-96] R. Canetti, U. Feige, O. Goldreich, and M. Naor, *Adaptively Secure Multi-Party Computation*, Proc. of 28th STOC, 1996, pp. 639–648.

[CKOR-97] R. Canetti, E. Kushilevitz, R. Ostrovsky, and A. Rosén, *Randomness vs. Fault-Tolerance*, Proc. of 16th PODC, 1997, pp. 35-44.

[CCD-88] D. Chaum, C. Crepeau, and I. Damgard, *Multiparty Unconditionally Secure Protocols*, Proc. of 20th STOC, 1988, pp. 11-19.

[CKu-89] B. Chor, E. Kushilevitz *A Zero-One Law for Boolean Privacy*, STOC 21 (1989) 62-72. Journal version in SIAM J. Disc. Math. 4 (1991) 36-47.

[CGK-90] B. Chor, M. Geréb-Graus, and E. Kushilevitz, *Private Computations Over the Integers*, FOCS 90, pp. 335-344. To appear in SICOMP.

[CGK-92] B. Chor, M. Geréb-Graus, and E. Kushilevitz, *On the Structure of the Privacy Hierarchy*, Journal of Cryptology, Vol. 7, No. 1, 1994, pp. 53-60.

[C-87] C. Crépeau, *Equivalence between Two Flavors of Oblivious Transfer*, Crypto 87.

[CK-88] C. Crépeau, J. Kilian *Achieving Oblivious Transfer Using Weakened Security Assumptions* , Proc. IEEE Symp. on Foundations of Computer Science, 1988.

[EGL-85] S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Comm. of ACM v. 28, 1985 pp. 637-647.

[FKN-94] U. Feige, J. Kilian, and M. Naor, *A minimal model for secure computation*, STOC 26 (1994), 554-563.

[FMR-85] M. Fischer, S. Micali, C. Rackoff *An Oblivious Transfer Protocol Equivalent to Factoring*, Manuscript.

[GHY-87] Z. Galil, S, Haber, and M. Yung, *Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model*, Crypto, 87.

[G-98] O. Goldreich, *Secure Multi-Party Computation*, unpublished manuscript, 1998. Available from ftp://theory.lcs.mit.edu/pub/people/oded/prot.ps

[GMW-87] O. Goldreich, S. Micali and A. Wigderson, *How to Play any Mental Game* , Proc. ACM Symp. on Theory of Computing, 1987.

[GV-87] O. Goldreich, and R. Vainish, *How to Solve any Protocol Problem – An efficiency Improvement*, Crypto 87.

[GL-90] S. Goldwasser, and L. Levin, *Fair Computation of General Functions in Presence of Immoral Majority*, Crypto 90.

[GMR-85] S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, Proc. ACM Symp. on Theory of Computing, pp. 291-304 1985.

[HM-97] M. Hirt and U. Maurer, *Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation*, Proc. of 16th PODC, 1997.

[IL-89] R. Impagliazzo and M. Luby, *One-way Functions are Essential for Complexity-Based Cryptography*, Proc. IEEE Symp. on Foundations of Computer Science, 1989.

[IR-89] R. Impagliazzo and S. Rudich, *On the Limitations of certain One-Way Permutations*, Proc. ACM Symp. on Theory of Computing, pp 44-61, 1989.

[K-88] J. Kilian, *Basing Cryptography on Oblivious Transfer* , Proc. ACM Symp. on Theory of Computing, pp 20-31, 1988.

[K-91] J. Kilian, *Completeness Theorem for Two-party Secure Computation* , Proc. ACM Symp. on Theory of Computing, 1991.

[Ku-89] E. Kushilevitz, *Privacy and Communication Complexity*, FOCS89, and SIAM Jour. on Disc. Math., Vol. 5(2), 1992, pp. 273-284.

[KOR-96] E. Kushilevitz, R. Ostrovsky, and A. Rosén, *Characterizing Linear Size Circuits in Terms of Privacy*, Proc. of 28th STOC, pp. 541–550, 1996.

[KOR-98] E. Kushilevitz, R. Ostrovsky, and A. Rosén, *Amortizing Randomness in Private Multiparty Computations*, Proc. of 17th PODC, pp. 81–90, 1998.

[KMO-94] E. Kushilevitz, S. Micali and R. Ostrovsky, *Reducibility and Completeness In Multi-Party Private Computations*, Proc. IEEE Symp. on Foundations of Computer Science, 1994, pp. 478-489.

[KR-94] E. Kushilevitz, and A. Rosén, *A Randomness-Rounds Tradeoff in Private Computation*, CRYPTO94, and SIAM Jour. on Disc. Math., Vol. 11(1), 1998, pp. 61-80.

[OVY-91] R. Ostrovsky, R. Venkatesan, and M. Yung. *Fair Games Against an All-Powerful Adversary*, extended abstract in the proceedings of Sequences '91, June 1991, Positano, Italy. See also *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* Vol 13. (Jin-Yi Cai ed.) pp. 155-169, 1993.

[RB-89] T. Rabin and M. Ben-Or, *Verifiable Secret Sharing and Multiparty Protocols with Honest Majority*, STOC 1989, ACM, pp. 73-85.

[R-81] M. Rabin, *How to Exchange Secrets by Oblivious Transfer*, TR-81 Aiken Computation Laboratory, Harvard, 1981.

[W-83] S. Weisner, *Conjugate Coding*, SIGACT News, Vol. 15, No. 2, pp. 78-88, 1983.

[Yao-82] A.C. Yao, *Protocols for Secure Computations*, Proc. of 23th FOCS, pp. 160-164, 1982.

[Yao-86] A.C. Yao *How to Generate and Exchange Secrets*, Proc. of 27th FOCS, pp. 162-167, 1986.

A n-private Protocols Using Embedded-OT Channels

In this appendix, we present an n-private protocol, that uses OT channels, to compute an arbitrary n-argument function f. Our starting point is the protocol presented in [GHY-87, GV-87, GL-90, BG-89] which also deals with malicious players. Here we assume that players are honest. This enables us to use a simplified version of the protocol, and prove Lemma 4.

Proof: The protocol goes as follows: we are given a circuit with addition and multiplication mod 2 gates, that computes the function f, the players do the following.

1 **Sharing the inputs:** Each player P_i shares each bit $x_{i,k}$ of its input x_i by choosing, uniformly, at random a vector $(a_1^{i,k},\ldots,a_n^{i,k})$ such that $\sum_{j=1}^{n} a_j^{i,k} = x_{i,k}$.[9] Each such $a_j^{i,k}$ is called a *share* of the secret $x_{i,k}$. The player P_i sends the share $a_j^{i,k}$ to P_j (over their common private channel[10]).

[9] All arithmetic operations in this protocol are modulo 2.

[10] Note that given an OT channel a private channel is easy to implement: for $\mathcal{S}$ to send a bit b to $\mathcal{R}$ he duplicates its bit twice, $\mathcal{R}$ chooses a selection bit arbitrarily and they execute their OT protocol.

2 **Evaluating the function:** The evaluation of the function is done in a bottom-up fashion. Each gate $c = a \circ b$ is evaluated using the shares corresponding to the inputs a and b of the gate. The evaluation ends with each player P_i holding a share c_i of the output c, where the vector of shares is uniformly distributed among the vectors whose sum is c. We distinguish between two cases according to the operation in the gate:

- **c=a+b:** P_i computes its share of c by summing its shares of a and b. I.e., $c_i = a_i + b_i$. (No interaction is needed.)
- **c=ab:** Note that $a \cdot b = (\sum_{i=1}^{n} a_i) \cdot (\sum_{j=1}^{n} b_j) = \sum_{1 \leq i,j \leq n} a_i \cdot b_j$. Each player P_i can compute (locally) $a_i b_i$. However, if player P_i will know $a_i \cdot b_j$ (for $j \neq i$) he might be able to compute b_j, violating the privacy requirement. Instead, we let P_i and P_j interact in a two-party protocol, so that at the end P_i will know $v_{i,j} \overset{\triangle}{=} (a_i \cdot b_j) - r_{i,j}$, and P_j will know $r_{i,j}$, where $r_{i,j}$ is a random bit (note that $v_{i,j} + r_{i,j}$ equals $a_i \cdot b_j$). This is done by letting P_j choose $r_{i,j}$ at random. Then, P_i receives from P_j, via their common OT-channel, the a_i-th element of the pair of values $((0 \cdot b_j) - r_{i,j}, (1 \cdot b_j) - r_{i,j})$ (this pair can be easily computed by P_j). Clearly, this element, is exactly $(a_i \cdot b_j) - r_{i,j}$, as desired. As they use the OT-channel, P_j has no idea which value P_i selected. We repeat this two-party protocol for each pair P_i, P_j. Each player computes $c_i = a_i \cdot b_i + \sum_{j \neq i} v_{i,j} + \sum_{j \neq i} r_{j,i}$. It can be verified that $c = \sum_{i=1}^{n} c_i$.

3 **Revealing $f(x_1, \ldots, x_n)$:** Each player P_i broadcasts its share of the output gate of the circuit. The sum of these shares is the desired value.

In the lemmas below, we verify (inductively) that during the computation each vector of shares has the required sum, and that the distribution in any proper subset of the shares is uniform. In addition, the interaction gives no information about previously computed shares. These properties give the correctness and privacy of the protocol. □

Let $\texttt{View}(T, \{x_i\}, \{R_i\}_{i \in T})$ denote the view that the set of players (coalition) T has on the communication, given that each player P_i $(1 \leq i \leq n)$ has input x_i and that each player P_i in T has random string R_i. This is a random variable which is determined by the choice of random strings R_i for all players P_i not in T. We include in this view only messages that goes from players in $\bar{T}$ to players in T. (Note that these messages together with the inputs and random strings of players in T completely define the messages sent among players in T and also messages sent from players in T to players in $\bar{T}$.)

In the above protocol there is no communication for addition gates. Hence the view consists only of messages received during the sharing stage, during the evaluation of multiplication gates, and during the revealing stage. The first claim says that in a single evaluation of a multiplication gate no information is revealed.

Claim 5 *Consider the subprotocol evaluating a gate $c = ab$. For all coalitions T, for all set of shares $(a_1, \ldots, a_n)$ $(b_1, \ldots, b_n)$ which are the input for this subprotocol (i.e., $\{a_i, b_i\}$ is the input for player P_i), for all choices of random strings for players in T, $\{R_i\}_{i \in T}$, and for all communication comm $\in \{0,1\}^s$, where $s = |T|(n - |T|)$, we have:*

$$\Pr[\texttt{View}(T, \{a_i, b_i\}, \{R_i\}_{i \in T}) = comm] = 2^{-s},$$

where the probability goes over all choices of R_i for $i \in \bar{T}$.

Proof: The communication that goes from $\bar{T}$ to T is as follows: for every $i \in T$ and $j \in \bar{T}$ the players P_i, P_j jointly "compute" $a_i b_j$ and $a_j b_i$. In computing $a_j b_i$ the player P_i does not get any message (his role is to pick a random $r_{j,i}$ and to send a messages over their common OT-channel). In computing $a_i b_j$ the player P_i receives a one bit message ($v_{i,j}$). Hence, the view of coalition T must be of size s. Moreover, as $r_{i,j}$ is chosen (by P_j) uniformly at random, then $v_{i,j}$ is also uniformly distributed in $\{0,1\}$ (independently of what a_i and b_j are). As all $r_{i,j}$'s are independent, the claim follows. □

The next claim shows that at each stage of the computation the vector of shares is uniformly distributed. This is particularly important in the revealing stage, when we need to be sure that only the output is revealed.

Claim 6 *Let $x_1,\ldots,x_n$ be an input to the protocol (i.e., x_i is the input for player P_i). Let C be a gate in the circuit and let c be the value of this gate when the input for the circuit is $x_1,\ldots,x_n$. Let $\vec{C}$ be a vector of shares that represents c in the above protocol. Then,*

- *$\sum_{i=1}^{n} C_i = c$ (correctness); and*
- *$\vec{C}$ is uniformly distributed among the vectors whose sum is c (privacy). I.e., let $c_1,\ldots,c_n$ satisfy $\sum c_i = c$ (there are 2^{n-1} such vectors) then $\Pr[\vec{C} = (c_1,\ldots,c_n)|\vec{x}] = 1/2^{n-1}$.*

Proof: The first part easily follows from the description of the protocol, by induction. The second part is also proved by induction. The claim is certainly true after the sharing stage (as this is the way the shares are chosen). Now suppose we evaluate a gate. If the gate is an addition gate, computing $C = A + B$, then

$$\begin{aligned}\Pr[\vec{C} = (c_1,\ldots,c_n)|\vec{x}] &= \sum_{a_1,\ldots,a_n;\sum a_i = A} \Pr[\vec{A} = (a_1,\ldots,a_n)|\vec{x}] \cdot \Pr[\vec{B} = (c_1 - a_1,\ldots,c_n - a_n)|\vec{x}] \\ &= 2^{n-1}\frac{1}{2^{n-1}}\frac{1}{2^{n-1}} = \frac{1}{2^{n-1}}.\end{aligned}$$

If the gate computes $C = A \cdot B$ then we can fix $\vec{A} = (a_1,\ldots,a_n)$ and $\vec{B} = (b_1,\ldots,b_n)$ and now show that for any such fixed choice still $\vec{C}$ satisfies the requirement. In particular, it suffices to show (by induction on i) that the probability that $C_1 = c_1,\ldots,C_i = c_i$, for $i \leq n-1$ is $1/2^i$. To do so, we consider the bits $r_{i,j}$ ($j \neq i$) and $r_{j,i}$ ($j \neq i$) and assign random values to each of them (that were not assigned values so far). As at least one of those random bits (e.g., $r_{n,i}$) is still "free" this implies that c_i will be uniformly distributed (as $r_{n,i}$ is one of the summands that construct c_i). Clearly, when we consider C_n all the random bits already got values and hence the value of C_n is already determined. □

We now turn to the proof of privacy of the whole protocol:

Claim 7 *For all coalitions T (of size $1 \leq |T| \leq n-1$), for all input $x_1,\ldots,x_n$, for all choices of random strings for players in T, $\{R_i\}_{i\in T}$, and for all possible communication comm*[11]

$$\Pr[\mathtt{View}(T, \{x_i\}, \{R_i\}_{i\in T}) = comm] = 2^{-d},$$

for $d = |T| \cdot n_{\bar{T}} + m \cdot |T| \cdot (n - |T|) + (n - |T| - 1)$, where m is the number of multiplication gates in the circuit for f, and $n_{\bar{T}}$ is the number of inputs for the circuit held by players in $\bar{T}$. (Again, the probability goes over all choices of R_i for $i \in \bar{T}$.)

Proof: In the sharing stage, each player in T receives a share (a bit) from each input to the circuit held by player in $\bar{T}$ (by definition there are $n_{\bar{T}}$ such bits). The properties of the secret-sharing guarantee that each of these bits is 0 with probability 1/2 and they are all independent. The evaluation of addition gates does not involve any communication. Claim 5 guarantees that in the evaluation of any multiplication gate, no matter what are the shares that the players start with, the view of the players in T consists of a random string of length $|T|(n - |T|)$. Also, note that each of these evaluations make use of new (independent)

[11] a communication is *possible* for $x_1,\ldots,x_n$ if it is consistent with $f(x_1,\ldots,x_n)$.

random bits. Finally if $f_1, \ldots, f_n$ are the shares representing the outcome of the circuit, then by Claim 6 this vector is uniformly distributed among the vectors whose sum equals $f(x_1, \ldots, x_n)$. Therefore, the players in T get in the revealing stage $n - |T|$ bits which form $2^{n-|T|-1}$ combinations each with equal probability. Note that if $|T| = n - 1$ then in the revealing stage the players in T get only one bit which is uniquely determined by $\vec{x}$. However, if $|T| < n - 1$ then the independence of the communication seen in the revealing stage and the communication seen in previous stages is guaranteed by the random bits $r_{i,j}$ for $i, j \in \bar{T}$. Combining all together we get the desired claim. □

Corollary 8 *For all coalitions T (of size $1 \leq |T| \leq n - 1$), for all inputs $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ such that $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ and such that $x_i = y_i$ for all $i \in T$, for all choices of random strings for players in T, $\{R_i\}_{i \in T}$, and for all communication comm*

$$\Pr[\texttt{View}(T, \{x_i\}, \{R_i\}_{i \in T}) = comm] = \Pr[\texttt{View}(T, \{y_i\}, \{R_i\}_{i \in T}) = comm],$$

where the probability goes over all choices of R_i for $i \in \bar{T}$.