Any Non-Private Boolean Function Is Complete For Private Multi-Party Computations

Eyal Kushilevitz* Silvio Micali†
Rafail Ostrovsky‡

TR-93-076
Novemeber, 1993

Abstract

Let g be an n-argument boolean function. Suppose we are given a *black-box* for g, to which n honest-but-curious players can secretly give inputs and it broadcasts the result of operating g on these inputs to all the players. We say that g is *complete* (for multi-party private computations) if for *every* function f, the n players can compute the function f n-privately, given the black-box for g. In this paper, we characterize the boolean functions which are complete: 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*. On the other hand, for non-boolean functions, we show that this two notions are *not* complementary. Our result can be viewed as a generalization (for multi-party protocols and for ($n \geq 2$)-argument functions) of the two-party case, where it was known that two-argument functions which contain "embedded-OR" are complete.

* Department of Computer Science, Technion. Research was partially 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.

‡ University of California at Berkeley Computer Science Division, and International Computer Science Institute at Berkeley. E-mail: `rafail@melody.berkeley.edu`. Supported by NSF postdoctoral fellowship and ICSI.

1 Introduction

We consider multi-party private computations. Quite informally, given an arbitrary 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 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 players – malicious (i.e. Byzantine) players and honest-but-curios players:

Secure Computation for malicious players. Malicious players can deviate from the prescribed protocol in an arbitrary manner, in order to violate the correctness and privacy constraints. The first general protocol 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] based on various assumptions (either intractability assumptions or the existence of private (untappable) communication channels between each pair of players). These solutions give t-privacy for $t < \frac{1}{2}$ or $t < \frac{1}{3}$ depending on the assumption made.

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 them may put together the information in their possession at the end of the protocol in order to infer additional information about the original individual inputs. It should be realized that in this honest-but-curios 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). Its importance also stems from compiler-type theorems, such as the one proved by [GMW-87] (with further extensions in many subsequent papers, for example, [BGW-88, CCD-88, RB-89]). Namely, there are algorithms transforming 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. In our setting, we do not put any computational restrictions on the power of the players, hence the notion of privacy is *information-theoretic*.[1] Information-theoretic privacy was examined by [BGW-88, CCD-88] which prove that every function is $n/2$-private, and was then the subject of considerable work (e.g., [CKu-89, CGK-92, BB-89, CGK-90]). Particularly, [CKu-89] have succeeded in characterizing the

[1] This is in oppose to other works, e.g, [GMW-87], that give *computational-privacy*. That is, some information is revealed in the information-theoretic sense, but this information cannot be extracted in polynomial-time.

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 f_i are 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* functions are not n-private even for the honest-but-curious players.

Completeness for secure computation. Sometimes, instead of giving an explicit t-private protocol for evaluating a function f, one can show that a protocol is *reducible* to an implementation of some other, simpler protocol. In [R-81, Yao-82, GMW-87] first such results were shown based on some assumptions. In particular, [GMW-87] gave the first such result for a multi-party protocols, assuming the existence of a one-way function. Without additional assumptions, a stronger result was established in [K-88, K-91] for two-party protocols: most generally, [K-91] shows that any two-argument function with an embedded-OR (to be discussed later) is complete for two-party secure computation. No result of this generality is, unfortunately, known for the multi-party scenario.

Our Contribution. We formally define the notion of *reducibility* among multi-party protocol problems. We say that f is reducible to g, if just by repeatedly using a black-box (or a trusted party) for computing g the n players can compute the value of f n-privately. That is, in any round of the computation, the players *secretly* supply arguments to the black-box and then the black-box *publicly* announces the result of operating g on these arguments. 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 is reducible to it. It can be seen that if g is complete then g itself cannot be n-private. As the definition requires that the same function g will be used for computing *all* functions f, it may be somewhat surprising that complete functions exist at all.

In this work we prove the existence of complete functions for multi-party n-private computations (in the honest-but-curious model.) Moreover, while previous research concentrated on finding a *single* complete function for multi-party protocols, 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 for all n-private protocols if and only if it is *not* n-private.

Our result thus shows a very strong dichotomy: every boolean function is either "simple enough" so it can be computed n-privately, or it is "sufficiently expressive" so that a black-box for it enables computing any function (not only boolean) n-privately. 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. For example, the *identity* function $ID(x_1, \ldots, x_n)$, which is 1 if all the inputs are equal to each other and 0 otherwise, is complete and can serve as g. Thus, in any model in which such a g may be

n-privately implemented (e.g., by using your favorite assumption) the computation of any other n-argument function can be implemented n-privately.

THE STRENGTH OF OUR RESULT. Our result is actually stronger, as we elaborate below:

- Although one can define the notion of reduction while allowing more complicated types of communication between players (such as private channels, broadcast channel etc.), it should be stressed that in our construction the only means of communication among the players is by interacting with the black-box (i.e., evaluating the function g).
- We consider the most interesting scenario where there are n players and both functions (i.e, the black-box function g and the function f that the players are trying to compute) are n-argument functions. This enables us to organize the computation in *rounds*, where in each round each player submits a value of a *single* argument to g (and the value of each argument is supplied by exactly one player).[2] Thus, no player is "excluded" at any round from 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.
- As mentioned, when we talk about privacy we do not put any computational restrictions on the power of the players; hence we achieve information-theoretic privacy. However, when we talk about protocols, we measure their *efficiency* in three ways: (1) in terms of the ratio between the expected running time of a Turing machine (or a circuit) that computes f versus the number of calls to g; (2) in terms of the computational complexity of computing inputs to g by each player; and (3) in terms of a parameter k (our protocol allows error probability of $2^{-O(k)}$). The protocol we introduce is efficient (polynomial) in all these measures.

Our main theorem gives a complete 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.

SUB-CONTRIBUTIONS. As mentioned, the special case of *two-party* computations and *two-argument* functions is implicit in previous works: [Ku-89, CKu-89] showed that if a two-argument function is not private then it contains an embedded-OR[3]. [K-91] showed that if a two-argument function g contains an embedded-OR then with a black-box for g it is possible to implement an *Oblivious Transfer* (OT)[4]. Finally, [K-88] showed that a black-box for OT is sufficient for computing any two-argument function privately; The combination

[2] Which player submits which argument is a permutation specified by the protocol.

[3] A function $g(i,j)$ contains an embedded-OR if $(\exists i_0, i_1, j_0, j_1, x_0, x_1)$ $(\forall a, b \in \{0,1\})$ $g(i_a, j_b) = x_{a \vee b}$.

[4] Oblivious transfer is protocol for two players: a Sender that holds two bit b_0 and b_1 and 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.

of these results gives our result for the special case $n = 2$ (i.e. only for two-party protocols and only for two-argument functions.) Our proof for general n is a generalization of the above argument, and each of its three components may be of independent interest:

1. We appropriately generalize the notion of embedded-OR for n-argument functions. We then show that if an n-argument function is not private then it contains an embedded-OR (this does *not* follow from the characterization of [CKu-89]).

2. We show that if an n-argument function g (boolean or not boolean) contains an embedded-OR then with a black-box for g it is possible to implement private channels between any two players.

3. We use a construction similar to this of [K-91] together with the private channels (we already implemented) to implement OT. It should be emphasized that OT in a multi-party setting has the additional requirement that listeners will not get any information. This additional requirement is handled by the implementation of the private channels. Finally, it follows from the work of [GHY-87, GV-87, BG-89] that a n-private computation of any function f can be implemented given private channels and OT. All together, our main theorem follows.

1.1 Organization of the paper

In section 2 we specify our model and definitions. In section 3 we prove our main lemma; In sections 4 and 5 we use the main lemma to implement private channels, and OT channels (respectively) between players; In section 6 we use the above constructions to prove our main theorem. Finally, section 7 contains a discussion of the results and some open problems. There are two appendices containing formal proofs.

2 Model and definitions

The system we consider is a collection of n synchronous, computationally unbounded players $P_1, P_2, \ldots, P_n$. All the communication between the players is done using a black-box for g, as described below.

Let f be an n-argument function defined over a finite domain. At the beginning of an execution, each party P_i has an input x_i taken from this domain. In addition, each party 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.[5] The black-box then publicly announces the result of evaluating the function g on the input messages.

[5] 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.

We allow the players to take "different seats" in different rounds. Formally, with each round i the protocol associates a permutation π_i.[6] The value computed by the black-box at round i, denoted s_i, is $s_i = g(m^i_{\pi_i(1)}, m^i_{\pi_i(2)}, \ldots, m^i_{\pi_i(n)})$.[7] Each message m^i_j, 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 if case of non-boolean f) announced by the black-box equals the value of $f(x_1, x_2, \ldots, x_n)$, with probability $\geq 1 - 2^{-O(k)}$, where k is a security parameter.

Let $\mathcal{F}$ be an n-party protocol, as described above. The *communication* $S(\vec{x}, \vec{r})$ 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 $r_1, \ldots, r_n$.

Definition 1 Let $\mathcal{F}$ be an n party protocol which computes a function f, and let T be a coalition of parties, $T \subseteq \{1, 2, \ldots, n\}$. We say that the 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 in their T entries (i.e. $\forall i \in T : x_i = y_i$) and for which f has the same value $f(\vec{x}) = f(\vec{y})$, for every choice of random inputs $\{r_i\}_{i \in T}$, and for every communication S

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

(The probability space is over the random inputs of all parties in $\bar{T}$.)

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 $\bar{T}$ (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 parties 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.

REMARK: In the above definition we require *perfect* privacy. That is, we require that the two distributions in the definition 1 are identical. We remark, that one can relax the above 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 (e.g., [GMW-87, BGW-88, CKu-89]).

[6] It follows from the construction that without loss of generality the sequence of permutations can be made oblivious at a price of $O(n^2)$. At a price of $O(n^4)$ it can even be made independent of the non-n-private function g!

[7] We assume here that the number of arguments of g is the same as the number of arguments of f (i.e., n) but this is not essential to the results. Also note that if g is a symmetric function, there is no need to permute the inputs to g.

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.

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 was then studied in many works (e.g., [R-81, FMR-85, OVY-90, IL-89, K-88, K-91]). The variant of OT protocol that we use here 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, K-91]).

Definition 4 *Oblivious Transfer (OT):* Let k be a security 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 must hold:

- $\mathcal{R}$ gets the value of b_c with probability greater then $1 - 2^{-O(k)}$, 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 random tapes of $\mathcal{S}$ and $\mathcal{R}$ respectively, and let $comm(b_0, b_1, c, r_{\mathcal{S}}, r_{\mathcal{R}}) \in \{0,1\}^{poly(k)}$ be a communication string. Then for all k and for all $c, b_0, b_1 \in \{0,1\}$ the following must hold :
$$Pr_{r_{\mathcal{S}}, r_{\mathcal{R}}}\left(\mathcal{R}(c, r_{\mathcal{R}}, comm(b_0, b_1, c, r_{\mathcal{S}}, r_{\mathcal{R}})) = b_c\right) \geq 1 - \frac{1}{2^{O(k)}}$$
- $\mathcal{R}$ 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 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 $comm$:
$$Pr_{r_{\mathcal{S}}}\left(comm \mid c, b_c, r_{\mathcal{R}}, b_{1-c} = 0\right) \;=\; Pr_{r_{\mathcal{S}}}\left(comm \mid c, b_c, r_{\mathcal{R}}, b_{1-c} = 1\right).$$
- $\mathcal{S}$ does not get any information about c. (In other words, $\mathcal{S}$ has the same view in the case where $c = 0$ and 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 $comm$:
$$Pr_{r_{\mathcal{R}}}\left(comm \mid b_0, b_1, r_{\mathcal{S}}, c = 0\right) \;=\; Pr_{r_{\mathcal{R}}}\left(comm \mid b_0, b_1, r_{\mathcal{S}}, c = 1\right).$$

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

3 A New Characterization of n-private functions

In this section we prove our main lemma. We start by showing that any non n-private function gives us a broadcast channel. Then we show that any function which is not n-private has two arguments such that when appropriately restricted to this two arguments any such function contains an embedded OR.

3.1 Establishing a Broadcast

Even if there are no privacy requirements, it is probably not obvious that a black-box computing a function g can be used for computing *any* function f, given that the black-box is the only means of communication. To see that this is possible it is enough to show that g can be used to implement a broadcast operation. In such a case each player can broadcast its input and then each of them can locally compute the value of the function.

Lemma 1 Broadcast channel is realizable given a black-box g, for any non-constant g.

If g is non-constant, it is easy to verify that there exist $a_1, \ldots, a_n$, and an index i such that

$$g(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq g(a_1, \ldots, a_{i-1}, \overline{a_i}, a_{i+1}, \ldots, a_n).$$

Therefore, we can let $n-1$ players supply the $n-1$ (fixed) input values $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$, and let the Sender supply the n-th value (a_i or $\overline{a_i}$). The value of this last argument alone determines whether a 0 or 1 will be broadcasted by the black-box g. ∎

3.2 Extracting embedded OR

We now proceed with our main lemma which establishes a new combinatorial characterization of all n-private functions.

First we present a generalized definition of what it means for an n-argument boolean function to have an "embedded-OR" and show that any two-argument function which is not 1-private contains an embedded OR. We then generalize this 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_1, x_2, y_1, y_2 and $\sigma \in \{0, 1\}$ such that $h(x_1, y_1) = h(x_1, y_2) = h(x_1, y_1) = \sigma$ but $h(x_2, y_2) = \bar{\sigma}$.

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

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

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 f_i are 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 is $\lceil n/2 \rceil$-private if and only if in every partition of the indices into two sets $S, \bar{S}$ each of size at most $\lceil n/2 \rceil$, the two-argument boolean function
$$g_S(\{x_i\}_{i \in S}, \{x_i\}_{i \in \bar{S}}) \triangleq g(x_1, \ldots, x_n)$$
is 1-private.

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

Lemma 2 (Main Lemma:) Let $g(x_1 \ldots, x_n)$ be any n-argument 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 and hence not 1-private. 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 such that g_S is not 1-private. For simplicity of notations we assume that n is even and $S = \{1, \ldots, n/2\}$. By Fact 2 g_S contains an embedded-OR. Hence, we have some inputs which form the following structure:

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

We will be able to finish the proof if we will show that it is possible to choose those inputs so that $u_i \neq v_i$ for exactly one i and $w_j \neq z_j$ for exactly one j. We will 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 one coordinate. All this process will maintain 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 be the set of indices on which u and v disagree (i.e., $u_k \neq v_k$). Define the following sets of vectors: T_m is the set of all vectors that can be obtained from u by replacing u_k in exactly m coordinates (in which $v_k \neq u_k$) by v_k. In particular, $T_0 = \{u\}$ and $T_{|L|} = \{v\}$. Also define a vector $x = (x_1, ..., x_{n/2})$ to be of type 1 if $g_S(x, w) = g_S(x, z)$ and of type 2 if $g_S(x, w) \neq g_S(x, z)$ (where w and z are the specific vectors we choose above). So in particular, u is of type 1 and v is of type 2.

We now claim that there must exist u', v' as required. Namely, one of them is of type 1, the other is of type 2 (i.e., u', v', w, z still form an OR-like structure), and they differ in exactly one coordinate. Suppose, towards a contradiction that this is not true. We will show

that this implies that for all $0 \leq m \leq |L|$ all the vectors in T_m are of type 1 contradicting the fact that v which is in $T_{|L|}$ is of type 2. The proof is by induction. It is true for $m = 0$ as T_0 contains only u which is of type 1. Suppose the induction is true for m. That is, all the vectors in T_m are of type 1. For each vector x in T_{m+1} there is a vector in T_m which differs from it in exactly one coordinate. Since we assumed that u', v' as above do not exist, this immediately implies that x is also of type 1. Since we reached a contradiction this implies the existence of such u', v'. Based on those, we can also show in a similar way the existence of w', z' as needed. The u', v', w', z' exhibit the fact that g contains an embedded-OR. ■

4 Private Computation implies Private Communication

In this section we prove that private channels can be simulated using a non-n-private black-box. This alone already gives some interesting corollaries, as well as some intuition for the simulation of the OT-channels by a black-box, which is given in the next section.

Lemma 3 Private point-to-point communication channels are realizable given a black-box g, for any non-n-private g (with error probability $2^{-O(k)}$).

Proof: We describe a protocol that uses a black-box for g and enables a Sender to send a bit B to the Receiver such that listeners get no information about the value of B. The key tool is that by Lemma 2 the Sender and Receiver can privately compute, using the black-box, OR of bits they hold (by feeding appropriate inputs into the special indices i, j guaranteed by Lemma 2 and letting the other players supply the other (fixed) inputs). A listener that listens to such a computation will know the outcome of the OR but will get no additional information about the input bits. The protocol goes as follows:

1. Sender chooses $m = O(k)$ bits $a_1, \ldots, a_m$ at random and Receiver chooses $b_1, \ldots, b_m$ at random. They use the black-box to compute bits $c_i = OR(a_i, b_i)$ for all $1 \leq i \leq m$.

2. Let S be the set of indices i such that $c_i = 1$. The Sender constructs a subset $E \subseteq S$ as follows: For every $i \in S$ if $a_i = 0$ then $i \in E$. If $a_i = 1$ then $i \in E$ with probability 1/2. Sender broadcasts (as in Lemma 1) the set E (broadcasting E is not necessary for the protocol but is used to simplify the analysis).

3. To send the bit B, the Sender broadcasts a set $D \subseteq E$ of the first k indices i such that $a_i = B$. If Sender does not have enough indices with the required property he aborts the protocol (this happens with exponentially in k small probability, where probability is taken over coin-tosses of Sender and Receiver).[8]

4. Receiver, upon receiving the set D checks the corresponding b_i's. If any of these b_i's is a zero, the Receiver decrypts the message as 1, otherwise it decrypts it as 0.

[8] Here, and in other places, the players can "restart" the protocol instead of aborting it. This will decrease the error probability but will make the efficiency to be *expected polynomial* and not *polynomial*.

To analyze the protocol, we first point out some simple properties it has.

- For every index i, $Pr(i \in S) = 3/4$ (the only case that i is not in S is if $a_i = b_i = 0$).

- $Pr(a_i = 1 | i \in S) = 2/3$, since $c_i = 1$ implies that (a_i, b_i) is one of $(0,1), (1,0), (1,1)$ each with equal probability. Formally, $Pr(a_i = 1 | i \in S) = Pr(a_i = 1 \wedge i \in S)/Pr(i \in S) = (2/4)/(3/4) = 2/3$.

- $Pr(a_i = 0 | i \in E) = 1/2$. To see this we restrict our attention to the conditional probability space where $i \in S$ (and note that $E \subseteq S$). Then we can write: $Pr(a_i = 0 | i \in E) = Pr(a_i = 0 \wedge i \in E)/Pr(i \in E)$. In this restricted space $Pr(a_i = 0 \wedge i \in E)$ equals $Pr(a_i = 0)$ which, as computed before, is exactly $1/3$. We can also write $Pr(i \in E) = Pr(i \in E | a_i = 0) \cdot Pr(a_i = 0) + Pr(i \in E | a_i = 1) \cdot Pr(a_i = 1)$. As computed $Pr(a_i = 0) = 1/3$ and $Pr(a_i = 1) = 2/3$. Also by the construction of E, $Pr(i \in E | a_i = 0) = 1$ and $Pr(i \in E | a_i = 1) = 1/2$. This gives us $Pr(i \in E) = 1 \cdot (1/3) + (1/2) \cdot (2/3) = 2/3$. Hence $Pr(a_i = 0 | i \in E) = (1/3)/(2/3) = 1/2$ as required. Making the probability of $a_i = 0$ and $a_i = 1$ equally likely was the goal of restricting ourselves to the set E.

- Assuming that the protocol is not aborted, then the chances that a "1" message is decrypted as a "0" equals the probability that for each of the k indices in D we have $b_i = 1$. This happens with probability 2^{-k}. (As $Pr(b_i = 1 | a_i = 1 \wedge c_i = 1) = 1/2$.) On the other hand a "0" message is *always* decrypted as 0 (As $Pr(b_i = 1 | a_i = 0 \wedge c_i = 1) = 1$.)

To show that the protocol is secure against listeners (i.e., that we indeed obtain a *private* channel), we need to show that the distribution of communication is *identical* in the case that the transmitted bit is 0 and the case that the bit is 1. Formally, let *comm* be any communication in the above protocol. Then we want to show that

$$Pr(comm | B = 0) \;=\; Pr(comm | B = 1).$$

Note that the communication in the above protocol may be of one of two forms:

- $c_1; c_2; \ldots; c_m; E$; abort ; or

- $c_1; c_2; \ldots; c_m; E; D$;

Also note that $c_1, \ldots, c_m$ and E are all constructed *independently* of what the value of B is. Now, since E has the property that $Pr(a_i = 0 | i \in E) = Pr(a_i = 1 | i \in E) = 1/2$ then (given $c_1, \ldots, c_m$ and E) the probability that less than k indices in E satisfy $a_i = B$ (in which case we abort the protocol) is the same for $B = 0$ and $B = 1$. Similarly, if there are at least k indices as needed, the probability of any D is the same no matter what the value of B is. ∎

REMARK: As the distribution in case that 0 is transmitted and the distribution in case that 1 is transmitted are identical, then we will also get the same distribution for all messages of t bits.
At this point we already get the following corollary:

Corollary 1 Let g be a non-n-private Boolean function. Given a black-box g, it is possible to compute any function f $\lfloor (n-1)/2 \rfloor$-privately.

Proof: We use the protocols of [BGW-88, CCD-88] that can compute any function f $\lfloor (n-1)/2 \rfloor$-privately, assuming the existence of point-to-point communication channels, and we simulate these channels using the above lemma. The simulation preserves the perfect privacy of these protocols, and introduces a small probability of error (bounded by the number of messages exchanged in the original protocols times 2^{-k}). ■

5 Constructing *Embedded Oblivious Transfer*

We have shown in our main lemma that any non n-private function f contains embedded OR on two arguments, with all the other arguments fixed to some pre-specified values. Thus, our basic strategy is for the two players who wish to communicate via an OT channel to supply the two remaining free arguments to f, while all the other players specify fixed arguments given by our main lemma. The resulting two-argument function is an OR function, which by the results of [K-91] is sufficient to implement OT between two players. However, there is a subtle difficulty in implementing a private OT-channel in a multi-player system which we must address: beside of the usual properties of an OT channel, we should guarantee that the information transmitted between the owners of the channel will not be revealed to the *listeners* (i.e. the other $n-2$ players). If OT is implemented as a black-box, then clearly no information is revealed to the listeners. However, since we implement OT using a black-box, which publicly announces each of its outcomes, we must also prove that no information is revealed to the listeners. In order to do so, we substitute every message which is being sent in a clear from Sender to Receiver by using a private channel (whose existence was proven in Lemma 3 using a non-n-private black-box!). The crucial fact which we establish is that when other players (listeners) hear the result of the conversation between the two players, they do not get *any* information about the OT channel[9]. Thus, every pair of players can realize a private OT channel from which the lemma follows. We now state and prove the lemma:

Lemma 4 Embedded OT-channel between players is realizable given a black-box g, for any non-n-private g.

[9] We remark that it is possible to implement a private OT channel directly, without using the private channel. The resulting protocol would be more efficient in terms of communication complexity but the proof that the protocol is private becomes much more involved. Thus, in this extended abstract, we preset a less efficient, but simpler proof.

Proof: We start with an OT protocol of [K-91, CK-88]. In order to show that it is secure again listeners as well (i.e. that it implements an Embedded OT-channel) we modify this protocol so that all communication between the Sender and the Receiver is done by using the *private channel* (which can be simulated by a black-box according to Lemma 3), or by computing OR where input bits of both the sender and the receiver are chosen at random (recall that, again, computing OR can be done using the black-box according to Lemma 2). ∎

For sake of completeness we present a complete protocol and its proof of security in appendix A. It is simpler than the one presented in [K-91].

6 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 $n-1$ "curious" players, assuming the existence of OT-channels, private channels and a broadcast channel. Such a protocol can be obtained by combining results from [GHY-87, GV-87, BG-89] (these works deal also with Byzantine players). Both the protocol and proof of security appear in appendix B. To conclude, this proves the following lemma:

Lemma 5 Given a broadcast channel, a point-to-point private communication channel and OT channel between each pair of processors is sufficient to implement n private protocol for any function 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. g is complete if and only if it is *not* n-private.

Proof:
($\Longrightarrow$) First, we show that any complete g can not 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 it). This however contradicts the results of [CKu-89] that shows that most functions are *not* n-private. It is important to note that this negative result of [CKu-89] does not depend on the running time of the protocol, and it allows a probability of error. [10]

[10] This impossibility result holds even if we allow the players to communicate not only using the black-box but also using broadcast channel and point-to-point communication channels.

($\Longleftarrow$) 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 we shown a protocol that can tolerate $n-1$ "curious" players, assuming the existence of OT-channels, private channels and a broadcast channel (Lemma 5). However, we have also shown how a black-box, computing *any* non-private function, can be used to simulate all these types of communication while preserving the privacy. In order to do so, we first proved in our main lemma (Lemma 2) that any non-private function can be reduced to an OR function on two arguments. Then, we have shown that based on our main lemma, we can implement all three primitives needed in Lemma 5. In particular, in Lemma 1 we show how to implement a broadcast channel; in Lemma 3 we show how to implement a private channel; and in Lemma 4 we show how to implement an OT channel. Combining these lemmas we get the result. (We remark that the proof of Lemma 4 utilizes Lemma 3.) ■

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

7 Conclusions and further extensions

7.1 Non-boolean functions

We have shown that *any* non-n-private boolean function g is complete. Namely, a black-box g can be used for computing any function f in a totally private way. Finally, let us turn our attention to non-boolean functions. Here, we can state the following lemma:

Lemma 6 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 simple: 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 1).

We now show that with no embedded-OR one can not compute an OR function. Assume, towards the contradiction, that we can, i.e., that there is some function f which does not have an embedded-OR, yet it could be used to compute an OR function. Since it can be used to compute an OR function, we can use it to implement OT (see appendix A). Hence, there exists an implementation of OT based on some f which does not have an embedded-OR. However, [K-91] have shown that for two-argument functions, only the ones that contain an embedded-OR, can be used to implement OT, deriving 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 1: A non-private function which does not contain an embedded-OR

7.2 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). It is an interesting question to characterize the *multi-output* functions g that are complete.

It is not clear how to extend the model and the results to the case of *Byzantine* 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, and hence by [BGW-88, CCD-88] can implement any f, $n/3$-privately with respect to Byzantine players.

If we turn our attention to polynomial-time players and protocols and relax the notion of privacy to hold only in a computational sense then it is possible to show that an implementation (which is computationally n-private) of any (information-theoretically) non-n-private complete function implies the existence of a one-way function (basically, since we have shown that it is equivalent to the implementation of OT, which implies a one-way function by [IL-89].) The best-known implementations of OT for polynomially-bounded players requires *trapdoor* one-way permutations [GMW-87], and [IR-89] have shown that if using black-box reductions, then implementing OT for polynomially-bounded players using one-way permutations (without trapdoor) is as difficult as separating $\mathcal{P}$ from $\mathcal{NP}$. Thus, using black-box reductions, *complete* functions are hard to implement (with computational privacy) without a trapdoor property. An interesting open question to study is the complexity assumptions needed to implement polynomial-time protocols for computing (with computational privacy) functions which are non-n-private and, at the same time, are *not* complete, as the results [IR-89] do not apply to this case, yet, the best known polynomially-bounded implementation still seems to require a trapdoor one-way permutation.

Acknowledgments

We wish to thank Oded Goldreich for helpful discussions. We thank Mihir Bellare for pointing out to us in 1991 that works of Chor, Kushilevitz and Kilian are complementary and thus imply a special case of our general result.

References

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

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

[B-86] Blum M., *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, S. Goldwasser *Multiparty Computation with Faulty Majority*, FOCS 1989.

[CCD-88] Chaum, D., 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.

[CGK-92] B. Chor, M. Geréb-Graus, and E. Kushilevitz, *On the Structure of the Privacy Hierarchy*, To appear in *Journal of Cryptology*.

[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.

[FMR-85] Fischer M., 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, 1987.

[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.

[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.

[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, No. 2, May 1992, pp. 273-284.

[OVY-90] 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 DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 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.

[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 APPENDIX: *Embedded Oblivious Transfer* protocol

In this appendix, we present a simplified version of the [K-91] protocol which works for honest but curious players and is secure against listeners as well. We first re-state the lemma.

Lemma 7 Embedded OT-channel between players is realizable given a black-box g, for any non-n-private g.

Recall that k is a security parameter and denote the two bits of the Sender by $\mathcal{B}_0$ and $\mathcal{B}_1$ and the selection bit of the Receiver by $\mathcal{I}$. At the end of the protocol, Receiver should get the value of $\mathcal{B}_{\mathcal{I}}$ and learns nothing about the bit $\mathcal{B}_{1-\mathcal{I}}$, while the Sender learns nothing about the selection bit $\mathcal{I}$. All the communication between the Sender and the Receiver (except for step (1) which is done directly using the black-box) is transmitted using their *private channel*). The protocol is as follows:

(0) The Sender selects two new *random* bits s_0 and s_1. The goal of steps (1)-(5) is that the Receiver will get one of them at *random* (i.e. Receiver will get a random $i \in \{0,1\}$ and the bit s_i) and will not have any information about the other bit s_{1-i}, while the Sender does not have any information concerning i. When this is completed, Receiver and Sender will be able to complete the protocol (step (6)) using s_0, s_1 and i.

(1) Sender chooses m $(= O(k))$ random bits $a_1, \ldots, a_m$; Receiver chooses $b_1, \ldots, b_m$ random bits. For each $1 \leq i \leq m$, Sender and Receiver execute $OR(a_i, b_i) = c_i$.

(2) Both players discard bits for which $OR(a_i, b_i) = 0$. In addition, for each of the remaining bits, Receiver discards each of the bits with $b_i = 1$ with probability $1/2$. He sends the set E of the discarded bits to the Sender (the goal of this is to make the probability of each remaining b_i to be 1 equal to its probability to be 0). The remaining bits are renumbered from 1 to n, where n is some constant fraction of the original m (if not the players abort the protocol).

(3) Receiver chooses $2k$ b's out of the remaining $b_1, \ldots, b_n$ such that exactly k of them are zero and k are one (if there is not enough such b's receiver aborts the protocol), and pairs them up into k distinct pairs of indices $\beta_\ell = (i_\ell, j_\ell)$, where in each pair $b_{i_\ell} \neq b_{j_\ell}$ and the order (which of the b's is 1 and which is 0) is chosen at random. Receiver sends the indices of the pairs $\beta_1, \ldots, \beta_k$ to Sender.

(4) For each pair β_ℓ, the Sender checks the corresponding bits a_{i_ℓ}, a_{j_ℓ}. If none of these pairs is $(1,1)$ he aborts the protocol (the probability of this event is 2^{-k}; the purpose of this will become clear in the proof of Claim 4). Otherwise, the Sender selects $k-1$ random bits $r_1, \ldots, r_{k-1}$. For each pair $\beta_\ell = (i_\ell, j_\ell)$, $1 \leq \ell < k$ he sends to Receiver a pair of bits $\alpha_\ell = (r_\ell \oplus a_{i_\ell},\ r_\ell \oplus s_0 \oplus s_1 \oplus a_{j_\ell})$. Additionally, using the remaining pair β_k Sender transmits a pair $\alpha_k = (r_1 \oplus \ldots \oplus r_{k-1} \oplus s_0 \oplus a_{i_k}\ ,\ r_1 \oplus \ldots \oplus r_{k-1} \oplus s_1 \oplus a_{j_k})$.

(5) For each pair β_ℓ, one of b's is 0 (in which case Receiver knows that the corresponding a must be a 1) and the other b is 1, which means that the corresponding a could be either zero or a one (each with probability 1/2). Therefore, for each $1 \leq \ell \leq k-1$, Receiver can recover the element of α_ℓ whose corresponding b is 0. Namely, he can compute either r_ℓ or $r_\ell \oplus s_0 \oplus s_1$ (but not both). Similarly, from the k-th pair α_k, Receiver can recover either $r_1 \oplus \ldots \oplus r_{k-1} \oplus s_0$ or $r_1 \oplus \ldots \oplus r_{k-1} \oplus s_1$ (but not both). Now, Receiver can take a XOR of the k recovered items. The XOR of the first $k-1$ items gives him either $r_1 \oplus \ldots \oplus r_{k-1}$ or $r_1 \oplus \ldots \oplus r_{k-1} \oplus s_0 \oplus s_1$. In addition, both elements of α_k contain $r_1 \oplus \ldots \oplus r_{k-1}$ therefore when he takes the xor of the k recovered items together, all the r_ℓ's cancel. So, from the first $k-1$ pairs, the contribution is either $s_0 \oplus s_1$ or nothing and from the last pair, the contribution is either s_0 or s_1. Hence, taking the "xor" of everything, yields either s_0 or s_1, and receiver can figure out, based on the order of the pairs, which case this is (it has the same probability of getting s_0 and s_1 as the order of b's is random).

(6) Finally, remember that in step (1), Sender selected two new random bits s_0 and s_1 which he sends to the Receiver such that the Receiver got one of them at random (i.e. the Receiver can compute an i and the bit s_i) and does not have any information about the other bit s_{1-i}, while the Sender does not have any information concerning i. Now, the Receiver requests Sender to send him two bits $\mathcal{B}_\mathcal{I} \oplus s_i$ and $\mathcal{B}_{1-\mathcal{I}} \oplus s_{1-i}$, where the *order* in which the Receiver requests them is randomly chosen (by the Receiver). Sender transmits the two bits requested by the Receiver. The point is that since the Receiver already knows s_i he can recover $\mathcal{B}_\mathcal{I}$, and since he has no information about s_{1-i} (and s_{1-i} was chosen at random), he also has no information about $\mathcal{B}_{1-\mathcal{I}}$. The sender, since he does not know i, i.e., which of s_0, s_1 was transmitted to the receiver (and since the order of the two bits requested was chosen at random) also has no idea which of $\mathcal{B}_\mathcal{I}, \mathcal{B}_{1-\mathcal{I}}$ the Receiver got.

Now we wish to verify that the protocol hides the value of $\mathcal{B}_{1-\mathcal{I}}$ (and of s_{1-i}) from the Receiver, and the value of $\mathcal{I}$ (and of i) from the Sender. Additionally, we need to show that information is protected from the listeners as well. Note however that only step (1) is done in the open, and all other steps are done using a private channel.

Claim 1 For any setting of $\mathcal{I}, \mathcal{B}_\mathcal{I}, \mathcal{B}_{1-\mathcal{I}}$ and $\mathcal{I}', \mathcal{B}'_\mathcal{I}, \mathcal{B}'_{1-\mathcal{I}}$, and for any communication string *comm* between Sender and Receiver, it is the case that for *listeners*:

$$Pr(comm|\mathcal{I}, \mathcal{B}_\mathcal{I}, \mathcal{B}_{1-\mathcal{I}}) \quad = \quad Pr(comm|\mathcal{I}', \mathcal{B}'_\mathcal{I}, \mathcal{B}'_{1-\mathcal{I}})$$

where the probability is taken over the coin-flips of both the Sender and Receiver.

Proof: For any setting of $\mathcal{I}, \mathcal{B}_\mathcal{I}, \mathcal{B}_{1-\mathcal{I}}$ we can generate a distribution which is *identical* (from listeners view) to the conversation above as follows:

- generate $c_1, \ldots, c_m$ as in step (1) of the protocol (which is independent of $\mathcal{I}, \mathcal{B}_{\mathcal{I}}, \mathcal{B}_{1-\mathcal{I}}$). That is, m bits each is chosen to be 1 with probability 3/4 and they are all independent.
- For each bit transmitted on the private channel in steps (2)-(6) (either from the Sender to the Receiver or vice-versa) generate a "transmission" from the distribution corresponding to the private channel (as in the proof of Lemma 3).

The crucial point to notice is that step (1) is simulated perfectly and all the subsequent steps are done using a private channel. However, each transmission on the private channel can be simulated *perfectly* by Lemma 3 and independently of the other transmissions, we are done. ∎

Next, we wish to show that in the above protocol, the Receiver gets no information about s_{1-i}. Recall that the view of the Receiver in the protocol consists of the messages it received. Namely, the m bits $c_1, \ldots, c_m$, where $c_i = OR(a_i, b_i)$, and the pairs $\alpha_1, \ldots, \alpha_k$ ($2k$ bits).

Claim 2 For all i (the index of the selected bit), for all values of the bits $b_1, \ldots, b_m$ and the pairs of indices $\beta_1, \ldots, \beta_k$, constructed by the Receiver according to the above protocol (in particular, the b_i's corresponding to each pair β_ℓ are different, and the order of the pairs implies that i is the selected bit), for all values s_0 and s_1, and for all $sender - comm \in \{0,1\}^{m+2k}$ consistent with the above values [11]

$$Pr[VIEW(Receiver, b_1, \ldots, b_m, \beta_1, \ldots, \beta_k, s_0, s_1) = sender - comm] = \frac{1}{2^{d+2k-1}},$$

where d is the number of b_i which are equal to 0 and where the probability is taken over all random choices of the Sender; namely, $a_1, \ldots, a_m$ and $r_1, \ldots, r_{k-1}$.

Proof: First, we consider the first m bits of the communication. As the communication is consistent with the b_i's then if $b_i = 1$ then so is $c_i = 1$, no matter what the value of the corresponding a_i is. On the other hand, for each of the d bits where $b_i = 0$, it must be that $a_i = c_i$, which happens with probability 1/2. Therefore the probability of $c_1, \ldots, c_m$, given the information of the receiver (particularly, $b_1, \ldots, b_m$) is 2^{-d}. Note that up to this point, we only assigned the values of a_i's whose corresponding b_i's equal 0.

Consider now the two bits t_1, t_2 in the communication by the sender, corresponding to the pair α_ℓ, for $1 \leq \ell \leq k-1$. We will show that the probability of having these two bits in the communication is 1/4. We first consider the case where $b_{i_\ell} = 0$ and $b_{j_\ell} = 1$ (the case $b_{i_\ell} = 1$ and $b_{j_\ell} = 0$ is handled similarly). In this case, the corresponding a_{i_ℓ} was already assigned the value 1. Hence to get this communication, we need $r_\ell \oplus a_{i_\ell} = t_1$ or

[11] consistency here means that for this choice of values, there exist random choices for the sender that leads for this communication by the sender. In particular, if the value of one of the b_i's equals 1 then the corresponding c_i must always be equal to 1. Also, the sum of elements in the pairs α_ℓ's corresponding to b_i's that equal 0, must be equal to the value of s_i (the correctness implies that no communication can be consistent with both $s_i = 0$ and $s_i = 1$).

alternatively, we need to fix $r_\ell = a_{i_\ell} \oplus t_1$ (note that t_1 is fixed, and that at this point a_{i_ℓ} is also fixed, and only r_ℓ is still "free"). This happens with probability 1/2. Now, to get $r_\ell \oplus a_{j_\ell} \oplus s_0 \oplus s_1 = t_2$, we need $a_{j_\ell} = r_\ell \oplus s_0 \oplus s_1 \oplus t_2$. Again, note that t_1, s_0 and s_1 are fixed, and that at this point we already fixed the value of r_ℓ, but a_{i_ℓ} is still "free". Hence a_{j_ℓ} get the required value with probability 1/2.

Finally, consider the last two bits t_1, t_2 in the communication, corresponding to the pair α_k. We will show that the probability of having these two bits in the communication is 1/2. First, we observe that the order in the first $k-1$ pairs $\beta_1, \ldots, \beta_{k-1}$ together with the value of i already determine the order in the pair β_k. Suppose we are in the case $b_{i_k} = 0$ and $b_{j_k} = 1$ (the case $b_{i_k} = 1$ and $b_{j_k} = 0$ is handled similarly). In this case, a_{i_k} was already assigned the value 1. Also, $r_1, \ldots, r_{k-1}$ were also assigned values, and s_0 and t_1 are fixed. Hence, we have no freedom here: it must be that $t_1 = r_1 \oplus \ldots \oplus r_{k-1} \oplus s_0 \oplus a_{i_k}$. Otherwise, the communication is inconsistent (as we assigned values to the random bit only if we had no choice). Now, to get $r_1 \oplus \ldots \oplus r_{k-1} \oplus s_1 \oplus a_{j_k} = t_2$, we need $a_{j_k} = r_1 \oplus \ldots \oplus r_{k-1} \oplus s_1 \oplus t_2$. Again, note t_1 and s_1 are fixed, and that at this point $r_1, \ldots, r_{k-1}$ are already fixed as well, so only a_{j_k} is still "free". Hence a_{j_k} get the required value with probability 1/2. [12]

Combining all together, and using the independence of all the random choices made by the Sender, the claim follows. ■

Informally, in the previous lemma, we have shown that the communication does not reveal any information about s_{1-i}. Now, we can show that the Receiver does not get any information about $\mathcal{B}_{1-\mathcal{I}}$ as well. (In other words, Receiver has the same view in the case where $\mathcal{B}_{1-\mathcal{I}} = 0$ and the case where $\mathcal{B}_{1-\mathcal{I}} = 1$):

Claim 3 For all k, for all $\mathcal{I}, \mathcal{B}_\mathcal{I} \in \{0,1\}$, for all random strings of Receiver $r_\mathcal{R}$ and sender $r_\mathcal{S}$ and for all communication $comm$:

$$Pr_{r_\mathcal{S}}\left(comm \mid \mathcal{I}, \mathcal{B}_\mathcal{I}, r_\mathcal{R}, \mathcal{B}_{1-\mathcal{I}} = 0\right) \;=\; Pr_{r_\mathcal{S}}\left(comm \mid \mathcal{I}, \mathcal{B}_\mathcal{I}, r_\mathcal{R}, \mathcal{B}_{1-\mathcal{I}} = 1\right).$$

Proof: From the previous lemma, we know that the probability of any communication string $comm$ according to the receivers view is the same up to step (5). But this implies that

$$Pr_{r_\mathcal{S}}\left(comm \mid i, s_i, r_\mathcal{R}, s_{1-i} = 0\right) \;=\; Pr_{r_\mathcal{S}}\left(comm \mid i, s_i, r_\mathcal{R}, s_{1-i} = 1\right)$$

In step (6), however, we xor $\mathcal{B}_{\mathcal{I}-1}$ with s_{i-1} which implies that the transmission of $\mathcal{B}_{\mathcal{I}-1} \oplus s_{i-1}$ is equally likely to be a 0 or a 1, from which the claim follows. ■

Next, we wish to show that sender does not get any information about $\mathcal{I}$. (In other words, that sender has the same view in the case where $\mathcal{I} = 0$ and the case where $\mathcal{I} = 1$). More formally:

[12] The distributions are not the same for $i = 0$ and $i = 1$; If $b_{i_k} = 1$ and $b_{j_k} = 0$ the order is flipped and so is the value of i. In such a case, we get a different distribution: t_2 is now already determined, and t_1 changes its value according to the random choice of a_{i_k}.

Claim 4 For all k, for all $\mathcal{B}_0, \mathcal{B}_1 \in \{0,1\}$, for all $r_{\mathcal{S}}$ and for all communication $comm$:

$$Pr_{r_{\mathcal{R}}}(comm \mid \mathcal{B}_0, \mathcal{B}_1, r_{\mathcal{S}}, \mathcal{I} = 0) \;=\; Pr_{r_{\mathcal{R}}}(comm \mid \mathcal{B}_0, \mathcal{B}_1, r_{\mathcal{S}}, \mathcal{I} = 1).$$

Proof: We break up the proof into two cases, depending on whether the protocol is aborted at step (4). In case it is aborted in step (4), the above claim follows trivially, as all the steps before the abortion are independent of $\mathcal{I}, \mathcal{B}_0, \mathcal{B}_1$.

Consider the case that the protocol is *not* aborted. The view of the sender can be divided into two parts: the first part,$comm_{1-5}$, its view during steps (1)-(5) of the protocol. This view consists of the m bits $c_i = OR(a_i, b_i)$; the set E of discarded bits sent at step (2); and the k pairs of indices β_ℓ that the receiver sends him at step (3). The other part, $comm_6$, its view during step (6), consists of the request issued at that step. First note that $comm_{1-5}$ is independent of $\mathcal{I}, \mathcal{B}_0, \mathcal{B}_1, s_0, s_1$ and i. We also claim that

$$Pr_{r_{\mathcal{R}}}(i = 0 | r_{\mathcal{S}}, comm_{1-5}) = \frac{1}{2}.$$

(Clearly, i is independent of $\mathcal{I}, \mathcal{B}_0$ and $\mathcal{B}_1$.) This is because, in case that the protocol is not aborted then for at least one pair β_ℓ the corresponding bits a_{i_ℓ}, a_{j_ℓ} are both 1. Therefore, the pair β_ℓ is ordered $(0,1)$ or $(1,0)$ each with probability $1/2$ (for pairs where one of the bits a_{i_ℓ}, a_{j_ℓ} is 1 and the other 0 the order of β_ℓ is uniquely determined), which says that i may be either 0 or 1 each with probability $1/2$ (to see this note that if we flip the order in this pair β_ℓ it flips the value of i but does not change the communication).

Based on this, $comm_6$ is either $\{\mathcal{B}_0 \oplus s_0, \mathcal{B}_1 \oplus s_1\}$ or $\{\mathcal{B}_0 \oplus s_1, \mathcal{B}_1 \oplus s_0\}$ (in a random order) with equal probability, no matter what $\mathcal{I}$ is. The claim follows. ■

B APPENDIX: The n-private protocol using private channels, broadcast and embedded-OT channels

In this appendix, we present an n-private protocol, that uses the types of channels that we can implement based on black-box. We start with the protocol presented in [GHY-87, GV-87, BG-89] which also deals with Byzantine players. Here we assume that players are honest. This enables us to use a simplified version of the protocol, and prove the following lemma:

Lemma 8 Given a broadcast channel, a point-to-point private communication channel and OT channel between each pair of processors is sufficient to implement n private protocol for any function f.

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

1. **Sharing the inputs:** Each player P_i shares its input x_i by choosing, uniformly, at random a vector $(a_1^i, \ldots, a_n^i)$ such that $\sum_{j=1}^n a_j^i = x_i$.[13] Each such a_j^i is called a *piece* of the secret x_i. The player P_i sends the piece a_j^i to P_j (over their common private channel).

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 pieces corresponding to the inputs a and b of the gate. The evaluation ends with each player P_i holding a piece c_i of the output c, where the vector of pieces 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 piece of c by summing its pieces of a and b. I.e., $c_i = a_i + b_i$. (No interaction.)
   - **c=ab:** In this case $c = a \cdot b = (\sum_{i=1}^n a_i) \cdot (\sum_{j=1}^n b_j) = \sum_{1 \le i,j \le n} a_i \cdot b_j$. 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 will 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 $v_{i,j} \triangleq (a_i \cdot b_j) - r_{i,j}$, 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 choosing $r_{i,j}$ 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 $c = \sum_{i=1}^n c_i$.

[13] All arithmetic operations are modulo 2.

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

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

Let $VIEW(T, \{x_i\}_{i \in T}, \{R_i\}_{i \in T})$ denote the view that the set of players (coalition) T has on the communication given that each player P_i in T has an input x_i and random string R_i. 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, for all choices of random strings for players in T, $\{R_i\}_{i \in T}$, and for all communication $comm \in \{0,1\}^s$

$$Pr\left[VIEW(T, \{a_i, b_i\}_{i \in T}, \{R_i\}_{i \in T}) = comm | a_1, \ldots, a_n, b_1, \ldots, b_n\right] = 2^{-s},$$

where $s = |T|(n - |T|)$, and 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 $i \in T, j \in \bar{T}$ 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 $comm$ 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. 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 $Pr[\vec{C} = (c_1, \ldots, c_n)] = 1/2^{n-1}$.

Proof: The first part is easily proved by induction. The second part is also proved by induction. It 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

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

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 $(1 \leq |T| \leq n-1)$, 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 $comm$[14]

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

where $d = (m+1)|T|(n - |T|) + (n - |T| - 1)$, and m is the number of multiplication gates in the circuit for f. (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 player in $\bar{T}$. 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 shares that the player 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) 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

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

determined. 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 2 *For all coalitions T $(1 \leq |T| \leq n-1)$, 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*

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

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