The All-or-Nothing Nature of Two-Party Secure Computation*

Amos Beimel[†] Tal Malkin[‡] Silvio Micali[§]

December 8, 1999

Abstract

A function f is computationally securely computable if two computationally-bounded parties, Alice, having a secret input x, and Bob, having a secret input y, can talk back and forth so that (even if one of them is malicious) (1) Bob learns essentially only $f(x, y)$ while (2) Alice learns essentially nothing.

We prove that, if *any* non-trivial function can be so computed, then so can *every* function. Consequently, the complexity assumptions sufficient and/or required for computationally securely computing f are the same for every non-trivial function f.

1 Introduction

SECURE COMPUTATION. Let f be a two-argument finite function, that is, $f : S_1 \times S_2 \rightarrow S_3$ (where S_1, S_2, and S_3 are finite sets), and let Alice and Bob be two possibly malicious parties, the first having a secret input $x \in S_1$ and the second having a secret input $y \in S_2$. Intuitively, securely computing f means that Alice and Bob keep turns exchanging message strings so that (1) Bob learns the value $z = f(x, y)$, but nothing about x (which is not already implied by z and y), no matter how he cheats, while (2) Alice learns nothing about y (and thus nothing about z not already implied by x), no matter how she cheats.

In a sense, therefore, a secure computation of f has two constraints: a *correctness constraint*, requiring that Bob learns the correct value of $f(x, y)$, and a *privacy constraint*, requiring that neither party learns more than he/she should about the other's input.

*A preliminary version of this paper appeared in *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 80–97. Springer, 1999.

[†]Dept. of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel. E-mail: beimel@cs.bgu.ac.il. This work was done while the author was a postdoctoral fellow in the Division of Engineering and Applied Sciences, Harvard University, supported by grants ONR-N00014-96-1-0550 and ARO-DAAL03-92-G0115.

[‡]Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology sq., Cambridge, MA 02139. E-mail: tal@theory.lcs.mit.edu. Supported by DARPA grant DABT63-96-C-0018.

[§]Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology sq., Cambridge, MA 02139.

Throughout this paper, any function to be securely computed is a finite, two-argument function.

The One-Sidedness of Secure Computation. The notion of secure computation informally recalled above is the traditional one used in the two-party, *malicious* model (cf., [GMW87, Section 4.2], and [Kil88, Kil90]). This notion is "one-sided" in that only Bob learns the result of computing f, while Alice learns nothing. Such one-sidedness is unavoidable in our malicious model. In principle, one could conceive of a more general notion of secure computation in which "both Alice and Bob learn $f(x, y)$."[1] However, such a more general notion is not achievable in a two-party, malicious model: the first party who gets the desired result, if malicious, may stop executing the prescribed protocol, thus preventing the other from learning $f(x, y)$.[2] Moreover, such a malicious party can terminate prematurely the execution of the prescribed protocol exactly when he/she "does not like" the result. For example, Cleve [Cle86] proves that two parties cannot flip a fair coin.

Trivial and Non-Trivial Functions. A function f is called *trivial* if it can be securely computed even if a cheating party has unlimited computational power, and *non-trivial* otherwise.

An example of a trivial function is the "projection of the first input;" namely the function $\mathrm{P}_1 : \{0,1\} \times \{0,1\} \to \{0,1\}$ so defined: $\mathrm{P}_1(b_0, b_1) = b_0$. Another example is the "exclusive-or function;" namely, the function $\mathrm{XOR} : \{0,1\} \times \{0,1\} \to \{0,1\}$ so defined: $\mathrm{XOR}(b_0, b_1) = b_0 + b_1 \bmod 2$. Indeed, a secure way of computing either function consists of having Alice send her secret bit to Bob. This elementary protocol clearly is a correct and private way of computing P_1. It also is a correct and private way of computing XOR. Indeed, Alice's revealing her own secret bit b_0 enables Bob to compute locally and correctly the desired XOR of b_0 and b_1. Moreover, Alice's revealing b_0 also satisfies the privacy constraint: Bob could deduce Alice's bit anyway from the output of the XOR function he is required to learn.

An example of a non-trivial function is the function $\mathrm{AND} : \{0,1\} \times \{0,1\} \to \{0,1\}$ so defined: $\mathrm{AND}(b_0, b_1) = b_0 \wedge b_1$. Another non-trivial function is the (chosen 1-out-of-2) oblivious transfer; namely, the function $\mathrm{OT} : \{0,1\}^2 \times \{0,1\} \to \{0,1\}$ so defined: $\mathrm{OT}((b_0, b_1), i) = b_i$, that is, Bob only learns the bit of Alice he chooses. (The non-triviality of these functions follows from [CK89].)

Secure Computability of Non-Trivial Functions. By definition, securely computing non-trivial functions is *conceivable* only when (at least one of) Alice and Bob are computationally bounded, but by no means *guaranteed*. Nonetheless, a series of results have established that secure computation of non-trivial functions is possible under complexity assumptions of various strengths. In particular,

- The OT function is securely computable under the assumption that integer factorization is computationally hard [Rab81, FMR84, EGL85, Cré88].[3]

[1] Or even a more general scenario where Bob learns $f(x, y)$ while Alice learns $g(x, y)$.

[2] Or $g(x, y)$ in the more general scenario.

[3] Rabin [Rab81] introduced a variant of the oblivious transfer, the *random oblivious transfer*, and provided an implementation of it which is provably secure in the honest-but-curious model. Fischer, Micali, and

- All functions are securely computable if factoring is hard [Yao86]; and, actually,
- All functions are securely computable if *any* trapdoor permutation exists [GMW87].[4]

Such results raise fundamental questions about the strength of the computational assumptions required for secure computation. In particular,

$Q1$: *What assumption is required for securely computing at least one non-trivial function?*

$Q2$: *What assumption is required for securely computing a given non-trivial function f?*

$Q3$: *Are there assumptions sufficient for securely computing some non-trivial function f but not sufficient for securely computing some other non-trivial function g?*

COMPLETENESS FOR SECURE COMPUTATION. Another important result is that the OT function is *complete* for secure computation [Kil88] (generalizing the analogous result of [GV87] in the honest-but-curious model).[5] By this we mean that, if OT is securely computable, then so are all functions. A bit more specifically, given any function f and any protocol securely computing OT, one can efficiently and uniformly construct a protocol securely computing f.

The completeness of the OT function raises additional fundamental questions. In particular,

$Q4$: *Are there other (natural) functions that are complete for secure computation?*

$Q5$: *Is there a (natural) characterization of the functions complete for secure computation?*

1.1 Main Results

A CHARACTERIZATION OF COMPLETE FUNCTIONS. In this paper we prove the following

> **Main Theorem:** *Any non-trivial function is complete for secure computation.*

Clearly, our result provides an explicit and positive answer to questions $Q4$ and $Q5$, and an explicit and negative answer to $Q3$. Our result also provides an *implicit* answer to questions $Q1$ and $Q2$. Namely, letting f be any given non-trivial function, and A_f be the assumption that f is securely computable:

> *For any non-trivial function g, assumption A_f is both necessary and sufficient for securely computing g.*

Rackoff [FMR84] improved his protocol so as to be provably secure against malicious parties. Even, Goldreich, and Lempel [EGL85] introduced the notion of the chosen 1-out-of-2 oblivious transfer, together with an implementation of it which is provably secure in the honest-but-curious model. Finally, Crépeau [Cré88] showed how to transform any secure protocol for the random oblivious transfer to a secure protocol for the chosen 1-out-of-2 oblivious transfer.

[4] The hardness of factoring implies the existence of trapdoor permutations, but the vice-versa might not hold.

[5] Kilian [Kil91] also proves a more general result, but in a different model, which we discuss in Section 1.2.

An Interpretation of Our Main Theorem. Our main theorem, combined with results from [IL89], implies that one-way functions are necessary for securely computing any non-trivial function (in [IL89] it is only proven that one-way functions are necessary for securely computing the OT function). It also suggests that just assuming the existence of one-way functions may be *insufficient* to guarantee secure computation. Let us explain. Impagliazzo and Rudich [IR89] show that, without also proving that $\mathcal{P} \neq \mathcal{NP}$, no protocol having oracle-access to a random function can be proved to compute the OT function securely. This result has been interpreted as providing strong evidence that "one-way functions are not sufficient for constructing a protocol securely computing the OT function." It is then according to the same interpretation that our main theorem suggests that, for *any* non-trivial function f, A_f should be stronger than the existence of one-way functions.

A Characterization of Trivial Functions. Is there a combinatorial property that makes a two-argument function securely computable by two, possibly malicious, parties with *unbounded* computing power (i.e., trivial)? In our paper we also provide such a characterization (actually crucial to the proof of our main theorem) in terms of *insecure minors*.[6]

We say that f contains an insecure minor if there exist inputs x_0, y_0, x_1, y_1 such that $f(x_0, y_0) = f(x_1, y_0)$ and $f(x_0, y_1) \neq f(x_1, y_1)$, and prove:

> **Main Lemma:** A two-argument function f is trivial if and only if f does not contain an insecure minor.

> **Main Corollary:** A two-argument function f is complete if and only if f contains an insecure minor.

1.2 Related Work

The Honest-but-Curious Model. Both completeness and characterization of non-trivial functions have been extensively investigated with respect to a weaker notion of two-party secure computation introduced in [GMW87]: the *honest-but-curious* model.[7] In this model, the parties are guaranteed to properly execute a prescribed protocol, but, at the end of it, each of them can use his/her own *view* of the execution to infer all he/she can about the other's input. In this model, because no protocol can be prematurely terminated, it is meaningful to consider "two-sided" secure computation of a function f; that is, one in which each party learns $f(x, y)$, but nothing else about the other's input that is not already implicit in $f(x, y)$ and his/her own input. Indeed this is the traditional notion of secure function computation in the honest-but-curious model.

Similar to the malicious model, a function is said to be trivial in the honest-but-curious model if it can be securely computed even if the two (honest-but-curious) parties have unbounded computing power, and non-trivial otherwise. The above mentioned results of [Yao86,

[6] Note that our main theorem provides a characterization of both trivial and non-trivial functions, though not a combinatorial-looking one!

[7] Originally called "the semi-honest model" in [GMW87].

GMW87] immediately imply that every two-argument function is securely computable in the honest-but-curious model, under the corresponding complexity assumptions (hardness of factoring and existence of trapdoor permutations).

A combinatorial characterization of the trivial functions in the honest-but-curious model was first given by Chor and Kushilevitz [CK89] for *Boolean* functions (i.e., predicates), and then by Kushilevitz [Kus89] for all functions.

While in the malicious model we prove that all non-trivial functions are complete, in the honest-but-curious one the "corresponding" theorem does not hold; there exists a (non-Boolean) function that is neither trivial nor complete [Kus89, Kil91, KKMO98].[8] On the other hand, Kilian, Kushilevitz, Micali, and Ostrovsky [KKMO98] prove that any non-trivial *Boolean* function is complete in the honest-but-curious model.

An Hybrid Model. Kilian [Kil91] characterizes the functions f that are complete in an *hybrid* model of secure computation. Namely, the functions f for which, given access to a *two-sided black-box* for f (i.e., one giving the result $f(x, y)$ to *both* Alice and Bob), one can construct, for any function, a *one-sided protocol* that is information-theoretically secure against unbounded malicious parties. He proves that these functions f are exactly those containing an embedded-or, a special case of our insecure minor (i.e., one satisfying the additional constraint $f(x_0, y_0) = f(x_0, y_1)$).

In sum, Kilian's result "reduces" standard (one-sided) protocols to two-sided black boxes. Notice that this is different (and not applicable) to our case, where we reduce standard protocols to standard protocols. (Indeed, our characterization of the complete functions is different, and there are functions that are complete in our setting but not in his.)

Also notice that two-sided black boxes might be implementable via "tamper-proof hardware" or in some other physical model, but, as explained above, *no protocol* can securely implement a two-sided black box for a function f against malicious parties.[9]

Reduction Models. *Black-box* reductions (as those of [Cré88, CK88, Kil91, KKMO98]) are an elegant way to build new secure protocols. While two-sided boxes are not implementable by secure protocols against malicious parties, one-sided black boxes can be (under certain complexity assumptions). Thus, one may consider completeness under one-sided black box reductions. Following our work, Kilian [Kil99] characterizes the functions that are complete under black-box reductions. This characterization implies, for instance, that the (non-trivial) OR function (complete in our computationally-bounded, malicious setting) is not complete under black-box reductions.[10] Thus, such reductions are not strong enough to

[8] [KKMO98] prove this by combining the following two results. [Kus89] shows an example of a function which is non-trivial yet does not contain an embedded or, and [Kil91] shows that a function that does not contain an embedded or cannot be complete in this model under black-box reductions (it is not known if this function is complete under our stronger notion of reductions). We note that this example is a function which contains an insecure minor, and thus *is* complete in the malicious (one-sided) model, as we prove in this paper.

[9] Two-sided boxes may instead be implemented by protocols (under certain complexity assumptions) in the honest-but-curious model.

[10] Indeed, in any protocol that uses a black-box for OR, the party receiving the output can input 0 to the black-box, thus obtaining the other party's input, without any way of being detected. Since this cannot be

solve the questions we are interested in. We thus use an alternative definition of a reduction that is natural for protocols secure against bounded malicious parties. Informally, for us a reduction is a transformation of a given secure *protocol* for f (rather than a one-sided black box for f) into a protocol for g secure against *computationally bounded* malicious parties.

Organization. In Section 2 we define protocols, and secure computation in the honest-but-curious model. In Section 3 we provide a definition of secure computation in the unbounded malicious model, and proceed to characterize the trivial functions. Finally, in Section 4 we characterize the complete functions, and prove that any non-trivial function is complete.

2 Preliminaries

2.1 Protocols

Following [GMR85], we consider a two-party protocol as a pair, (A, B), of Interactive Turing Machines (ITMs for short). Briefly, on *input* (x, y), where x is a private input for A and y a private input for B, and *random input* (r_A, r_B), where r_A is a private random tape for A and r_B a private random tape for B, protocol (A, B) computes in a sequence of rounds, alternating between A-rounds and B-rounds. In an A-round (B-round) only A (only B) is active and sends a message (i.e., a string) that will become an available input to B (to A) in the next B-round (A-round). A computation of (A, B) ends in a B-round in which B sends the empty message and computes a private *output*.[11]

TRANSCRIPTS, VIEWS, AND OUTPUTS. Letting E be an execution of protocol (A, B) on input (x, y) and random input (r_A, r_B), we make the following definitions:

- The *transcript* of E consists of the sequence of messages exchanged by A and B, and is denoted by $\mathrm{TRANS}^{A,B}(x, r_A, y, r_B)$;
- The *view of* A consists of the triplet (x, r_A, t), where t is E's transcript, and is denoted by $\mathrm{VIEW}_A^{A,B}(x, r_A, y, r_B)$;
- The *view of* B consists of the triplet (y, r_B, t), where t is E's transcript, and is denoted by $\mathrm{VIEW}_B^{A,B}(x, r_A, y, r_B)$;
- The *output of* E consists of the string z output by B in the last round of E, and is denoted by $\mathrm{OUT}_B(y, r_B, t)$, where t is E's transcript.

In all the above the superscript (A, B) will sometimes be omitted when clear from the context. (Note that the superscript "A, B" is omitted in our notation for the output OUT_B because it only depends on B's view. Thus, though the transcript t may depend on what A does, once t is determined, B's output is determined by t and B's input and coin tosses.)

prevented, any protocol which is unconditionally secure using a one-sided OR black-box can be transformed into an unconditionally secure protocol, implying that only trivial functions can be black-box reduced to OR.

[11] Due to the one-sidedness of secure computation, only machine B produces an output.

We consider the three random variables $\mathrm{TRANS}(x, \cdot, y, r_B)$, $\mathrm{TRANS}(x, r_A, y, \cdot)$, and $\mathrm{TRANS}(x, \cdot, y, \cdot)$, respectively obtained by randomly selecting r_A, r_B, or both, and then outputting $\mathrm{TRANS}(x, r_A, y, r_B)$. We also consider the similarly defined random variables $\mathrm{VIEW}_A(x, \cdot, y, r_B)$, $\mathrm{VIEW}_A(x, r_A, y, \cdot)$, $\mathrm{VIEW}_A(x, \cdot, y, \cdot)$, $\mathrm{VIEW}_B(x, \cdot, y, r_B)$, $\mathrm{VIEW}_B(x, r_A, y, \cdot)$, and $\mathrm{VIEW}_B(x, \cdot, y, \cdot)$.

Polynomial-Time Protocols. A protocol (A, B) is called *polynomial time* if there is a fixed polynomial P such that, in every execution in which the length of both private inputs is $\leq k$, the number of steps taken by both A and B in that execution is $\leq P(k)$.

Security parameters. If k is a positive integer, we denote by 1^k the unary representation of k (i.e., the string consisting of k 1-symbols). We say that an execution of a protocol (A, B) has *security parameter* k if the private input of A is of the form $(1^k, x)$ and the private input of B is of the form $(1^k, y)$. (Thus, de facto 1^k is a "common input" while x and y are the "real private inputs.")

As we shall see, the security parameter controls the amount of privacy in the computationally-bounded model of secure computation. Also, because our protocols securely compute finite functions, if they are polynomial time and executed with security parameter k, then, as k grows, they take a number of steps that is polynomial in k alone.

2.2 Secure Computation in the Honest-but-Curious Model

Though we are investigating fundamental properties of secure computation in the malicious model, as an intermediate step we shall also consider secure computation in the much simpler honest-but-curious model. In the latter model, if parties Alice and Bob execute the protocol (A, B), then it is guaranteed that Alice scrupulously follows the instructions of A and that Bob will do the same with respect to B. In such conditions, it is easy to enforce the correctness condition (for securely computing a function f), but not necessarily the privacy conditions: either party may in fact try to deduce as much information possible from its own view of an execution of (A, B) about the other's private input.

Formal definitions follow, both in the case where the parties are computationally unbounded and in the case where they are bounded. In the latter case a security parameter is needed, but, as we anticipated, this will not affect the notion of correctness, but only that of privacy (which will hold in a computational sense, rather than in an information theoretic one).

2.2.1 The Unbounded Case

Definition 1 *Let $f : S_1 \times S_2 \to S_3$ be a finite function. A protocol (A, B)* securely computes f against unbounded honest-but-curious parties, *if the following conditions hold:*

1. Correctness: $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B$, *letting* $v = \mathrm{VIEW}_B^{A,B}(x, r_A, y, r_B)$,

$$\mathrm{OUT}_B(v) = f(x, y).$$

2. Privacy:

Alice's Privacy: $\forall x_0, x_1 \in S_1$, $\forall y \in S_2$, $\forall r_B$, *if* $f(x_0, y) = f(x_1, y)$ *then*

$$\mathrm{VIEW}_B^{A,B}(x_0, \cdot, y, r_B) = \mathrm{VIEW}_B^{A,B}(x_1, \cdot, y, r_B).^{12}$$

Bob's Privacy: $\forall x \in S_1$, $\forall y_0, y_1 \in S_2$, $\forall r_A$,

$$\mathrm{VIEW}_A^{A,B}(x, r_A, y_0, \cdot) = \mathrm{VIEW}_A^{A,B}(x, r_A, y_1, \cdot).$$

2.2.2 The Bounded Case

In this model, we relax the above definition by requiring the privacy conditions to hold when the (curious) parties are bounded to polynomial time computation. That is, the probability distributions of the privacy condition are taken to be polynomial-time indistinguishable, rather than equal. For the sake of completeness, we recall the definition of indistinguishable distributions [GM82, Yao82a], and spell out the definition of secure computation in the bounded honest-but-curious model.

Definition 2 *Two sequences of probability distributions,* $\{V_k\}_{k=1}^{\infty}$ *and* $\{W_k\}_{k=1}^{\infty}$*, are* polynomial-time indistinguishable*, denoted* $\{V_k\}_{k=1}^{\infty} \approx \{W_k\}_{k=1}^{\infty}$*, if for every probabilistic polynomial-time Turing Machine* M*, every integer* $c \geq 1$ *and for every sufficiently large* k*,*

$$\left| \Pr_{v \in V_k}\left[M\left(v, 1^k\right) = 1\right] - \Pr_{w \in W_k}\left[M\left(w, 1^k\right) = 1\right] \right| \leq \frac{1}{k^c}.$$

Definition 3 *Let* $f : S_1 \times S_2 \to S_3$ *be a finite function. A polynomial time protocol* (A, B) securely computes f against bounded honest-but-curious parties*, if the following conditions hold:*

1. Correctness: $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B$, *letting* $v = \mathrm{VIEW}_B^{A,B}(\,(1^k, x), r_A, (1^k, y), r_B\,)$,

$$\mathrm{OUT}_B(v) = f(x, y).$$

2. Privacy:

Alice's Privacy: $\forall x_0, x_1 \in S_1$, $\forall y \in S_2$, $\forall r_B$, *if* $f(x_0, y) = f(x_1, y)$ *then*

$$\left\{\mathrm{VIEW}_B^{A,B}(\,(1^k, x_0), \cdot, (1^k, y), r_B\,)\right\}_{k=1}^{\infty} \approx \left\{\mathrm{VIEW}_B^{A,B}(\,(1^k, x_1), \cdot, (1^k, y), r_B\,)\right\}_{k=1}^{\infty}.$$

Bob's Privacy: $\forall x \in S_1$, $\forall y_0, y_1 \in S_2$, $\forall r_A$,

$$\left\{\mathrm{VIEW}_A^{A,B}(\,(1^k, x), r_A, (1^k, y_0), \cdot\,)\right\}_{k=1}^{\infty} \approx \left\{\mathrm{VIEW}_A^{A,B}(\,(1^k, x), r_A, (1^k, y_1), \cdot\,)\right\}_{k=1}^{\infty}.$$

[12] Equivalently, the corresponding transcripts are identically distributed (and similarly below).

3 A Combinatorial Characterization of Trivial Functions

So far, we have intuitively defined a trivial function to be one that is computable by a protocol that is *secure against unbounded malicious parties*.[13] Combinatorially characterizing trivial functions, however, requires first a quite formal notion of secure computation in our setting, a task not previously tackled. This is what we do below.

3.1 Secure Computation in the Unbounded Malicious Model

In this model Alice or Bob may be malicious, namely cheat in an arbitrary way, not using the intended ITM A (or B), but rather an arbitrary (computationally unbounded) strategy A' (or B') of their choice. The definition of secure computation in the malicious model requires some care. For example, it is not clear how to define what the input of a malicious party is.

We handle the definition of secure computation in the spirit of [MR92] and [DMR99] (though their definition is aimed at secure computation in the multi-party scenario of [BGW88, CCD88]). Intuitively, we require that when Alice and Bob are honest then Bob computes the function f correctly relative to his own input and Alice's input. We also require that, if Bob is honest, then, for any possible malicious behavior of Alice, Bob computes the function f correctly relative to his own input and Alice's input: a string defined by evaluating a predetermined input function on Alice's view of the joint computation. Because the computation is one-sided and a malicious Bob might not output any value, the correctness requirement is limited to the above two cases. Finally, we require privacy for an honest Alice against any possibly malicious Bob, and privacy for an honest Bob against any possibly malicious Alice.

Definition 4 *Let $f : S_1 \times S_2 \to S_3$ be a finite function. A protocol (A, B)* securely computes f against unbounded malicious parties, *if the following conditions hold:*

1. Correctness: $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B,$

When both Alice and Bob are honest:
Letting $v = \mathrm{VIEW}_B^{A,B}(x, r_A, y, r_B)$, then $\mathrm{OUT}_B(v) = f(x, y)$.

When only Bob is honest: For every strategy A' there is a mapping $\mathcal{I}_{A'} : \{0,1\}^* \to S_1$ such that, letting

$$v'_{A'} = \mathrm{VIEW}_{A'}^{A',B}(x, r_A, y, r_B) \text{ and } v'_B = \mathrm{VIEW}_B^{A',B}(x, r_A, y, r_B),$$

$$\mathrm{OUT}_B(v'_B) = f(\mathcal{I}_{A'}(v'_{A'}), y).^{14}$$

[13]By this we do not mean that the parties participating in a protocol computing a trivial function are computationally-unbounded, but that the "privacy and correctness" of their computation holds even when one of them is allowed to be malicious and computationally-unbounded.

[14]Note that a malicious strategy A' may ignore input x altogether. (In particular, some A' may execute the protocol according to a different input x', which may or may not depend on x.) We essentially use input x here in order to use our established notation.

Also note that the correctness condition when both parties are honest implies that the mapping $\mathcal{I}_A$ (i.e., the mapping for honest Alice) always produces a "correct" input, that is either x itself or an input "equivalent" to x, in the sense that it yields the same output $f(x, y)$.

2. Privacy:

Honest Alice's Privacy: For every strategy B', $\forall x_0, x_1 \in S_1$, $\forall y \in S_2$, $\forall r_B$, *if*

$$f(x_0, y) = f(x_1, y)$$

then

$$\mathrm{VIEW}_{B'}^{A,B'}(x_0, \cdot, y, r_B) = \mathrm{VIEW}_{B'}^{A,B'}(x_1, \cdot, y, r_B).$$

Honest Bob's Privacy: For every strategy A', $\forall x \in S_1$, $\forall y_0, y_1 \in S_2$, $\forall r_A$,

$$\mathrm{VIEW}_{A'}^{A',B}(x, r_A, y_0, \cdot) = \mathrm{VIEW}_{A'}^{A',B}(x, r_A, y_1, \cdot).$$

Note that security against malicious parties implies security against honest-but-curious parties. That is,

Fact 1 *If a protocol securely computes the function f in the unbounded malicious model, then it also securely computes f in the unbounded honest-but-curious model.*

Definition 5 (Trivial and non-trivial functions) *A finite function f is called* trivial *if there exists a protocol securely computing it in the unbounded malicious model; otherwise, f is called* non-trivial.

3.2 The Combinatorial Characterization

We prove that the trivial functions are exactly those that do not contain an insecure minor (a simple generalization of an *embedded or* [CK89]).[15]

Definition 6 (Insecure minor) *A function $f : S_1 \times S_2 \to S_3$ contains an insecure minor if there exist $x_0, x_1 \in S_1$, $y_0, y_1 \in S_2$, and $a, b, c \in S_3$ such that $b \neq c$, and $f(x_0, y_0) = f(x_1, y_0) = a$, $f(x_0, y_1) = b$, and $f(x_1, y_1) = c$. Graphically,*[16]

$\mathbf{f}:$	$\mathbf{x_0}$	$\mathbf{x_1}$
$\mathbf{y_0}$	a	a
$\mathbf{y_1}$	b	c

[15] An embedded or is an insecure minor in which $a = b$. As shown in [CK89], having an embedded or implies non-triviality in the two-sided *honest-but-curious* model, and characterizes the *Boolean* non-trivial functions in this model.

[16] This graphical convention will be used in the rest of the paper, namely a table where columns correspond to possible inputs for Alice, rows correspond to possible inputs for Bob, and the entries are the corresponding output values.

Protocol (A, B)
A**, on input** $x \in S_1$**:**
Send to Bob the message $a \stackrel{\text{def}}{=} f(x, \hat{y})$.
B**, on input** $y \in S_2$**, upon receipt of the message** a **from Alice:**
Let $T \stackrel{\text{def}}{=} \{e \in S_1 \ : \ f(e, \hat{y}) = a\}$.
If $T \neq \emptyset$, find the lexicographically first element, $\hat{e}$, of T and set $\mathrm{OUT}_B(y, r_B, a) \stackrel{\text{def}}{=} f(\hat{e}, y)$.
Else, if $T = \emptyset$, set $\mathrm{OUT}_B(y, r_B, a) \stackrel{\text{def}}{=} f(\hat{x}, y)$.

Figure 1: A secure protocol (against unbounded malicious parties) for a function f not containing an insecure minor.

Examples. As immediately apparent from their tables, the function AND consists of an insecure minor, and the function OT contains several insecure minors – one of which is highlighted below. (In both cases these insecure minors actually are embedded ors: indeed, for Boolean functions, every insecure minor is an embedded or.)

AND :	**0**	**1**
0	0	0
1	0	1

OT :	$(\mathbf{0},\mathbf{0})$	$(\mathbf{0},\mathbf{1})$	$(\mathbf{1},\mathbf{0})$	$(\mathbf{1},\mathbf{1})$
0	0	0	*1*	*1*
1	0	1	*0*	*1*

Furthermore, consider the function $\mathrm{MIN} : \{2, 3\} \times \{1, 3\} \to \{1, 2, 3\}$ defined below. This function consists of an insecure minor which is not an embedded or.

MIN :	**2**	**3**
1	1	1
3	2	3

Theorem 1 *A function $f(\cdot, \cdot)$ is trivial if and only if f does not contain an insecure minor.*

Proof. Theorem 1 follows from the following Claim 1 and Claim 2.

Claim 1 If f does not contain an insecure minor then it is trivial.

Proof. We assume that f does not contain an insecure minor and prove that f is trivial by constructing a protocol, (A, B), described in Fig. 1, that securely computes f against malicious unbounded parties. In protocol (A, B), $\hat{x}$ and $\hat{y}$ are arbitrary, fixed elements of, respectively, S_1 and S_2.

We first prove the correctness of the protocol when both Alice and Bob are honest (and thus have x and y as their respective inputs). Notice that honest Bob is guaranteed to find $\hat{e}$ because T is non empty, as it must contain honest Alice's input x. Moreover, because

f does not contain an insecure minor, we must have $f(x,y) = f(\hat{e},y)$ (otherwise $x, \hat{e}, \hat{y}, y$ would constitute an insecure minor). Thus, honest Bob's output, $f(\hat{e},y)$, is correct.

To prove (A,B)'s correctness when only Bob is honest, we define the following input function $\mathcal{I}_{A'} : \{0,1\}^* \to S_1$. If T is empty, then $\mathcal{I}_{A'}(x, r_A, a) \stackrel{\text{def}}{=} \hat{x}$; otherwise, $\mathcal{I}_{A'}(x, r_A, a)$ is defined to be the lexicographically first element of T. Then, it always holds that $\mathrm{OUT}_B(y, r_B, a) = f(\mathcal{I}_{A'}(x, r_A, a), y)$ and correctness follows. (Notice that the input function $\mathcal{I}_{A'}$ is the same for every adversary A'.)

Let us now prove Honest Alice's privacy. Let B' be any adversary strategy for Bob, and x_0, x_1, and y be such that $f(x_0,y) = f(x_1,y)$. Then, for any random input, r_A, for honest Alice, and any random input, r_B, for B', we have

- $VIEW_{B'}^{A,B'}(x_0, r_A, y, r_B) = (y, r_B, t_0)$, where, by the protocol, the transcript $t_0 = f(x_0, \hat{y})$
- $VIEW_{B'}^{A,B'}(x_1, r_A, y, r_B) = (y, r_B, t_1)$, where, by the protocol, the transcript $t_1 = f(x_1, \hat{y})$.

Honest Alice's privacy then follows because, from the fact that f does not contain an insecure minor, we must have $t_0 = t_1$ (otherwise, $x_0, x_1, y, \hat{y}$ would constitute an insecure minor). Therefore, the above two views of B' are identical.

Honest Bob's privacy follows immediately from the fact that (A,B) is a one-round protocol in which Bob sends no information to any possibly malicious Alice, and thus her view is clearly the same for any input Bob may have. □

Let us now prove the second part of Theorem 1.

Claim 2 If a function f is trivial then it does not contain an insecure minor.

Proof. If f is trivial, then, by definition, there is some protocol, (A,B), securely computing f against unbounded parties. In particular, by Fact 1, Protocol (A,B) securely computes f against honest-but-curious unbounded parties. Assume now, for sake of contradiction, that f contains an insecure minor, x_0, x_1, y_0, y_1; namely, that there are $a, b, c \in S_3$ such that $b \neq c$ and

$\mathbf{f}:$	$\mathbf{x_0}$	$\mathbf{x_1}$
$\mathbf{y_0}$	a	a
$\mathbf{y_1}$	b	c

By Bob's privacy from Definition 1, for every r_A,

$$\mathrm{VIEW}_A(x_0, r_A, y_1, \cdot) = \mathrm{VIEW}_A(x_0, r_A, y_0, \cdot).$$

By ranging over all possible r_A we get

$$\mathrm{TRANS}(x_0, \cdot, y_1, \cdot) = \mathrm{TRANS}(x_0, \cdot, y_0, \cdot).$$

On the other hand, by Alice's privacy, since $f(x_0, y_0) = f(x_1, y_0) = a$, for every r_B,

$$\mathrm{VIEW}_B(x_0, \cdot, y_0, r_B) = \mathrm{VIEW}_B(x_1, \cdot, y_0, r_B).$$

Again, by ranging over all possible r_B we get

$$\text{TRANS}(x_0, \cdot, y_0, \cdot) = \text{TRANS}(x_1, \cdot, y_0, \cdot).$$

Finally, again by Bob's privacy

$$\text{TRANS}(x_1, \cdot, y_0, \cdot) = \text{TRANS}(x_1, \cdot, y_1, \cdot).$$

Thus, by transitivity,

$$\text{TRANS}(x_0, \cdot, y_1, \cdot) = \text{TRANS}(x_1, \cdot, y_1, \cdot). \tag{1}$$

Equation (1) states that two transcripts are identically distributed though f's output is different (b in one case and c in the other). If Bob's output were dependent only on the transcript we would have already contradicted the correctness requirement. However, recall that Bob's output depends on his entire view, which includes his random input. Thus, we should prove that the above transcripts are identically distributed for every (fixed) random input of Bob. To this end, we use the following proposition of [CK89], which holds for every protocol. Informally, this proposition says that, if changing the inputs for both Alice and Bob yields the same transcript, then changing the input for Alice only (or Bob only) also yields the same transcript.

Proposition 1 ([CK89]) Let $u_0, u_1, v_0, v_1, r_{A,0}, r_{A,1}, r_{B,0}, r_{B,1}$ be inputs and random inputs and t be a transcript such that

$$\text{TRANS}(u_0, r_{A,0}, v_0, r_{B,0}) = \text{TRANS}(u_1, r_{A,1}, v_1, r_{B,1}) = t.$$

Then,
$$\text{TRANS}(u_0, r_{A,0}, v_1, r_{B,1}) = \text{TRANS}(u_1, r_{A,1}, v_0, r_{B,0}) = t.$$

Note that the above proposition implies that for every transcript, the set of inputs and random inputs of Alice and Bob that yield that transcript is a Cartesian product of two smaller sets. Formally,

Proposition 2 For every protocol (A, B) and transcript t, let

$$Q_A(t) \stackrel{\text{def}}{=} \{(x, r_A) \ : \ \exists y', r'_B \ \text{TRANS}(x, r_A, y', r'_B) = t\},$$

$$Q_B(t) \stackrel{\text{def}}{=} \{(y, r_B) \ : \ \exists x', r'_A \ \text{TRANS}(x', r'_A, y, r_B) = t\},$$

and

$$Q_{A,B}(t) \stackrel{\text{def}}{=} \{(x, r_A, y, r_B) \ : \ \text{TRANS}(x, r_A, y, r_B) = t\}.$$

Then,

$$Q_{A,B}(t) = Q_A(t) \times Q_B(t).$$

Now, by Proposition 2, it holds that

$$\Pr_{r_A,r_B}[\mathrm{TRANS}(x_0,r_A,y_1,r_B)=t\ ] = \Pr_{r_A}[(x_0,r_A)\in Q_A(t)]\cdot\Pr_{r_B}[(y_1,r_B)\in Q_B(t)]\,, \tag{2}$$

and similarly

$$\Pr_{r_A,r_B}[\mathrm{TRANS}(x_1,r_A,y_1,r_B)=t\ ] = \Pr_{r_A}[(x_1,r_A)\in Q_A(t)]\cdot\Pr_{r_B}[(y_1,r_B)\in Q_B(t)]\,. \tag{3}$$

Notice that Equation (1) implies that the left-hand sides of Equations (2) and (3) are equal, and thus that

$$\Pr_{r_A}[(x_0,r_A)\in Q_A(t)] = \Pr_{r_A}[(x_1,r_A)\in Q_A(t)]\,. \tag{4}$$

Equation (4) and Proposition 2 imply that for every r_B and t,

$$\Pr_{r_A}[\mathrm{TRANS}(x_0,r_A,y_1,r_B)=t] \;=\; \Pr_{r_A}[\mathrm{TRANS}(x_1,r_A,y_1,r_B)=t]\,. \tag{5}$$

In fact, if r_B and t are such that $(y_1,r_B)\notin Q_B(t)$, then both sides of Equation (5) are 0. Else, if $(y_1,r_B)\in Q_B(t)$, then the left-hand side (respectively, right-hand side) of Equation (5) equals the left-hand side (respectively, right-hand side) of Equation (4). Now, since Equation (5) holds for every r_B and t, we conclude that

$$\forall r_B\quad \mathrm{TRANS}(x_0,\cdot,y_1,r_B)=\mathrm{TRANS}(x_1,\cdot,y_1,r_B). \tag{6}$$

Recall that the view of Bob is defined as his input, his random input, and the transcript of the communication. By Equation (6), for every r_B the communication transcript between Alice and Bob is identically distributed when the respective inputs of Alice and Bob are x_0 and y_1, and when their respective inputs are x_1 and y_1. Because in both cases Bob has the same input y_1 and the same random input r_B, his view is identically distributed in both cases, namely:

$$\forall r_B\quad \mathrm{VIEW}_B(x_0,\cdot,y_1,r_B)=\mathrm{VIEW}_B(x_1,\cdot,y_1,r_B). \tag{7}$$

Equation (7) contradicts the correctness requirement of Definition 1, because $f(x_0,y_1)=b\neq c=f(x_1,y_1)$, whereas the identical distributions of Bob's view imply that Bob has the same output distribution in both cases. This concludes the proof of Claim 2. □

Claim 1 and Claim 2 complete the proof of Theorem 1. ■

3.3 The Round Complexity of Secure Computation against Unbounded Malicious Parties

Typically, multiple rounds and probabilism are crucial ingredients of secure computation. As stated in the following corollary, however, two-party secure computation in the unbounded malicious model is an exception.

Corollary 1 *If a function f is securely computable in the unbounded malicious model, then it is so computable by a deterministic single-round (actually, single-message) protocol.*

Proof. The corollary follows immediately from our proof (rather than the statement) of Theorem 1. Our proof, in fact, shows that, if a function f is securely computable in the unbounded two-party malicious model, then it is so computed by the protocol of Fig. 1, which is deterministic and calls for only one message to be sent (from A to B). ■

Together with the above corollary, our proof of Theorem 1 (actually, of Claim 2 alone) also immediately implies the following relationship between secure computation in the unbounded honest-but-curious model and in the unbounded malicious one.

Corollary 2 *For every two-argument function f, one of the following holds: Either*

1. *f is securely computable deterministically and in one round in the unbounded malicious model; or*
2. *f is not securely computable in the unbounded honest-but-curious model, even by probabilistic and multi-round protocols.*

4 Characterization of Complete Functions

In this section we prove that every non-trivial function (i.e., any function containing an insecure minor) is *complete* for secure computation in the computationally bounded malicious model. This model is both adequate and realistic: a malicious party may deviate from its prescribed protocol in any way it wants, as long as it uses a polynomial-time adversarial strategy. (Even malicious parties cannot perform an exponential amount of computation!)

4.1 Secure Computation in the Bounded Malicious Model

To characterize the trivial functions, in Section 3 we started by defining secure computation in the unbounded malicious model. Similarly, to characterize the complete functions, we could start by formally defining the notion of secure computation in the bounded malicious model (that we have already informally discussed).

The latter task, however, is rather complex (see, for instance, [Gol98] for some of the subtleties involved) and goes well beyond the scope of this paper. Fortunately, to prove our characterization, we only need two basic properties of secure computation in bounded malicious model, that are indeed satisfied by any reasonable definition (thus relieving us from choosing which definition should be considered the "right one").

The First Property. The first property is the following analogue to Fact 1 from the unbounded model:

Fact 2 *If a protocol securely computes the function f in the bounded malicious model, then it also securely computes f in the bounded honest-but-curious model.*

The Second Property. The second property relies on the "protocol compiler" of [GMW87]. In essence, Goldreich, Micali and Wigderson:

1. provide a polynomial-time transformation mapping any two-party protocol P into a two-party protocol P_{GMW};
2. provide a specific (intuitive) definition of secure computation in the bounded malicious model; and
3. assuming the existence of a bit-commitment scheme, show that if a protocol P securely computes a function f in the bounded honest-but-curious model, then P_{GMW} securely computes (according to their definition) the same f in the bounded malicious model.

Our second property states that (as universally accepted in the field) P_{GMW} actually securely computes f according to any reasonable definition of secure computation in the bounded malicious model.[17] That is,

Fact 3 ([GMW87]) *Assuming that a secure bit-commitment exists, if a protocol P securely computes the function f in the bounded honest-but-curious model, then P_{GMW} securely computes f in the bounded malicious model.*

4.2 Reductions and Completeness

As usual, the definition of completeness relies on that of a *reduction.* Informally, a function g reduces to a function f if a secure protocol for f can be converted to a secure protocol for g without any additional assumptions.

Definition 7 (Reductions) *Let $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ be finite functions. We say that the function g* reduces *to f in the bounded malicious model (respectively, in the bounded honest-but-curious model) if there exists a transformation[18] T_g such that, if P is a protocol securely computing f in the bounded malicious model (respectively, in the bounded honest-but-curious model), then $T_g(P)$ is a protocol securely computing g in the same model.*

Definition 8 (Completeness) *A function $f(\cdot,\cdot)$ is* complete *for bounded malicious secure computations (respectively, bounded honest-but-curious secure computations) if every finite function $g(\cdot,\cdot)$ reduces to f in the bounded malicious model (respectively, in the bounded honest-but-curious model).*

Our reductions immediately satisfy the following basic and natural properties.

Lemma 1 *Let f and g be finite functions such that g reduces to f in the bounded malicious model of secure computation. Then, any assumption sufficient for securely computing f in the bounded malicious model is also sufficient for securely computing g in the same model.*

[17] For any protocol (A, B), the GMW compiler "forces" a player to follow the prescribed protocol. That is, any player P, when sending a message m, also provides a zero knowledge proof [GMR85] that m is indeed the message that, if honest, P would send at the *same point* of the prescribed protocol with the *same history* so far. Their compiler works because (1) zero-knowledge proofs work in the bounded malicious model (i.e., they are a special type of secure computation in that model); (2) zero-knowledge proofs exist for any NP-statement [GMW86]; and (3) [GMW87] reduces proving that one is following a prescribed polynomial-time protocol to proving an NP-statement.

[18] All reductions presented in this paper consist of efficient and uniform transformations.

Lemma 2 *Let f, g, and h be finite functions. If h reduces to g and g reduces to f then h reduces to f.*

Lemma 3 *Let f and g be any finite functions. If g can be computed securely in the bounded malicious model without any assumptions then g reduces to f in the bounded malicious model.*

As mentioned in the introduction, our reductions are not black-box ones, but are natural and very suitable for investigating which assumptions are sufficient for secure computation. In contrast, black-box reductions are not strong enough to establish our main theorem. For instance, the (non-trivial) OR function is not complete under black-box reductions [Kil99] (see Footnote 10). Hence, the latter reductions do not give any indication about the minimal assumption necessary for computing OR securely.

We also note that, by [Yao86, GMW87], if factoring is hard or if trapdoor permutations exist, then all finite functions (even the trivial ones!) are complete.

4.3 Main Theorem

Theorem 2 *If $f(\cdot,\cdot)$ is a non-trivial function, then f is complete in the bounded malicious model.*

Proof. As an intermediate step, we start by proving the theorem for the bounded honest-but-curious model (which was defined in Definition 3).

Claim 3 If $f(\cdot,\cdot)$ is a non-trivial function, then f is complete in the bounded honest-but-curious model.

Proof. In Theorem 1, we have proved that a function f is non-trivial if and only if it contains an insecure minor. Further, it is proven in [GV87] that OT is complete in the bounded honest-but-curious model. Therefore, to establish our claim it suffices to prove that whenever f contains an insecure minor then OT reduces to f in the bounded honest-but-curious model. This is what we show below.

Let (A_f, B_f) be a secure protocol computing the function f in the bounded honest-but-curious model. Because the function f contains an insecure minor, there are values x_0, x_1, y_0, y_1, a, b and c such that $b \neq c$, $f(x_0, y_0) = f(x_1, y_0) = a$, $f(x_0, y_1) = b$, and $f(x_1, y_1) = c$.

Consider now the protocol $(A_{\mathrm{OT}}, B_{\mathrm{OT}})$ of Fig. 2, which uses both the above insecure minor and protocol (A_f, B_f). We shall prove that $(A_{\mathrm{OT}}, B_{\mathrm{OT}})$ securely computes the function OT.

By the very construction of $(A_{\mathrm{OT}}, B_{\mathrm{OT}})$, it holds that $z_\imath = f(x_{\beta_\imath}, y_1)$. Therefore, $z_\imath = b$ if $\beta_\imath = 0$, and $z_\imath = c \neq b$ otherwise. This implies that the output of B_{OT} is correct. We next argue that the privacy constrains are satisfied for bounded honest-but-curious parties A_{OT} and B_{OT}. First note that the only messages exchanged in $(A_{\mathrm{OT}}, B_{\mathrm{OT}})$ are during the executions of (A_f, B_f). Since (A_f, B_f) computes f securely, A_f (and thus A_{OT}) does not learn any information about $\imath$. Recall that B_f is not allowed to learn any information that is not implied by his input and the output of the function. In the case of OT, this means

Protocol $(A_{\rm OT}, B_{\rm OT})$	
$A_{\rm OT}$'s **input:** $\beta_0, \beta_1 \in \{0,1\}$	
$B_{\rm OT}$'s **input:** $\imath \in \{0,1\}$	
$A_{\rm OT}$'s and $B_{\rm OT}$'s **code:**	Execute protocol (A_f, B_f) on input $x_{\beta_0}, y_{\bar\imath}$. Denote by z_0 the output of B_f. Execute protocol (A_f, B_f) on input $x_{\beta_1}, y_\imath$. Denote by z_1 the output of B_f.
$B_{\rm OT}$'s **output:** If $z_\imath = b$ then output 0, else output 1.	

Figure 2: A secure protocol in the bounded honest-but-curious model for computing OT from a function f containing an insecure minor with values $x_0, x_1, y_0, y_1, a, b, c$.

that $B_{\rm OT}$ should not learn any information about $\beta_{\bar\imath}$. However, the only information that $A_{\rm OT}$ sends that depends on $\beta_{\bar\imath}$ are during the execution of (A_f, B_f) on input $(x_{\beta_{\bar\imath}}, y_0)$ and, thus, $z_{\bar\imath} = a$ for both values of $\beta_{\bar\imath}$. By the fact that (A_f, B_f) computes f securely, B_f does not learn any information on $\beta_{\bar\imath}$. □

Remarks. Note that protocol $(A_{\rm OT}, B_{\rm OT})$ is secure only if $B_{\rm OT}$ is honest.

Also note that in protocol $(A_{\rm OT}, B_{\rm OT})$ it is crucial that only B_f gets the outputs z_0 and z_1 of subroutine (A_f, B_f). That is, if $A_{\rm OT}$ gets z_0 or z_1 then she can learn $B_{\rm OT}$'s input for at least one of the possible values of her input (since either $b \neq a$ or $c \neq a$ or both).

Finally, note that in our proof of Claim 3, we only used a black-box reduction. Thus, the completeness of any non-trivial function in the honest-but-curious model holds also with respect to black-box reductions.

Let us now prove an "hybrid result" bridging the completeness in the two bounded models of secure computation.

Claim 4 Let $f(\cdot,\cdot)$ be any finite function. If f is complete in the bounded honest-but-curious model then it is complete in the bounded malicious model.

Proof. Let f be a complete function in the bounded honest-but-curious model, and for any finite function h let $T_h^{\rm hbc}$ be a transformation from any protocol securely computing f in the honest-but-curious model into a protocol securely computing h in the same model. We need to prove that a similar transformation exists for the bounded malicious model. That is, we need to prove that, for any finite function g, there exists a transformation T_g mapping any protocol securely computing f in the bounded malicious model into a protocol securely computing g in the same model.

Let $P = (A_f, B_f)$ be a protocol securely computing f in the bounded malicious model, and let g be any finite function. On input P, the transformation T_g proceeds as follows.

First, T_g applies transformation T_g^{hbc} to P so as to yield the resulting protocol $(A_g^{\mathrm{hbc}}, B_g^{\mathrm{hbc}})$. The latter protocol securely computes g in the honest-but-curious model because of our hypothesis on T_g^{hbc}, and because, by Fact 2, P securely computes f also in the bounded honest-but-curious model.

Second, T_g applies transformation $T_{\mathrm{OT}}^{\mathrm{hbc}}$ to P, so as to yield the resulting protocol $(A_{\mathrm{OT}}^{\mathrm{hbc}}, B_{\mathrm{OT}}^{\mathrm{hbc}})$. For exactly the same reasons above, $(A_{\mathrm{OT}}^{\mathrm{hbc}}, B_{\mathrm{OT}}^{\mathrm{hbc}})$ securely computes the function OT in the bounded honest-but-curious model.

Third, T_g applies to protocol $(A_{\mathrm{OT}}^{\mathrm{hbc}}, B_{\mathrm{OT}}^{\mathrm{hbc}})$ the construction of [IL89] to obtain a one-way function F.[19]

Fourth, T_g uses the result of [HILL99] to transform F into a pseudo-random (in the sense of [BM82, Yao82]) generator G.

Fifth, T_g uses the result of [Nao89] to transform G into a bit-commitment protocol BC.

Finally, due to Fact 3, T_g uses BC and the compiler of [GMW87] to transform $(A_g^{\mathrm{hbc}}, B_g^{\mathrm{hbc}})$ into a protocol (A_g, B_g) that securely computes g in the bounded malicious model. This completes the proof of Claim 4.[20] □

Claims 3 and 4 together yield the desired proof of Theorem 2. ■

Acknowledgments

We thank Joe Kilian and Eyal Kushilevitz for insightful comments. We also thank Lior Pachter for risking his life to save a preliminary version of this work.

References

[Blu82] M. Blum. Coin flipping by phone. *IEEE Spring COMPCOM*, pages 133–137, 1982.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proc. of the 20th Symp. on Theory of Computing*, pages 1–10, 1988.

[BM82] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. on Computing*, 13:850–864, 1984. Preliminary version in *FOCS '82*, 1982.

[CCD88] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proc. of the 20th Symp. on Theory of Computing*, pages 11–19, 1988.

[19] [IL89] provides an explicit transformation mapping any protocol that securely computes OT in the bounded honest-but-curious model into a one-way function.

[20] An alternative proof of Claim 4 could be obtained by using the results of [GHY88] instead of [GMW87, Nao89].

[CK88] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *Proc. of the 29th IEEE Symp. on Foundations of Computer Science*, pages 42–52, 1988.

[CK89] B. Chor and E. Kushilevitz. A zero-one law for Boolean privacy. *SIAM J. on Discrete Math.*, 4(1):36–47, 1991. Preliminary version in *STOC '89*, 1989.

[Cle86] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. of the 18th ACM Symp. on the Theory of Computing*, pages 364–369, 1986.

[Cré88] C. Crépeau. Equivalence between two flavors of oblivious transfers. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer-Verlag, 1988.

[DMR99] Y. Dodis, S. Micali, and P. Rogaway. Concurrent reducibility for information-theoretically secure computation. Manuscript, 1999.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.

[FMR84] M. J. Fischer, S. Micali, and C. Rackoff. A secure protocol for the oblivious transfer. Presented in EUROCRYPT '84, 1984. Printed version in *J. of Cryptology*, 9(3):191–195, 1996.

[GHY88] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 135–155. Springer-Verlag, 1988.

[GM82] S. Goldwasser and M. Micali. Probabilistic Encryption. In *J. of Computer and System Sciences* , 28(2):270–299, 1984. Preliminary version in *STOC '82*, 1982.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, 18:186–208, 1989. Preliminary version in STOC '85, 1985.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems. In *J. of the ACM*, 38(1):691–729, 1991. Preliminary version in *FOCS '86*, 1986.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th Symp. on the Theory of Computing*, pages 218–229, 1987.

[Gol98] O. Goldreich. Secure multi-party computation (working draft). Available from http://www.wisdom.weizmann.ac.il/~oded/foc.html, 1998.

[GV87] O. Goldreich and R. Vainish. How to solve any protocol problem – an efficiency improvement. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 73–86. Springer-Verlag, 1988.

[Has90] J. Hastad. Pseudo-random generators under uniform assumptions. In. *Proc. of the 22nd ACM Symp. on Theory of Computing*, pages 395-404, 1990.

[HILL99] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. *SIAM J. on Computing*, 28(4):1364–1396, 1999. This is the journal version of [ILL89, Has90].

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proc. of the 30th IEEE Symp. on Foundations of Computer Science*, pages 230–235, 1989.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random number generation from one-way functions. In *Proc. of the 21st ACM Symp. on the Theory of Computing*, pages 12–24, 1989.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proc. of the 21st ACM Symp. on the Theory of Computing*, pages 44–61, 1989.

[Kil88] J. Kilian. Basing cryptography on oblivious transfer. In *Proc. of the 20th ACM Symp. on the Theory of Computing*, pages 20–31, 1988.

[Kil90] J. Kilian. *Uses of Randomness in Algorithms and Protocols.* MIT Press, 1990.

[Kil91] J. Kilian. A general completeness theorem for two-party games. In *Proc. of the 23th ACM Symp. on the Theory of Computing*, pages 553–560, 1991.

[Kil99] J. Kilian. (More) general completeness theorems for two-party games. Manuscript, 1999.

[KKMO98] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. on Computing*, 1998. To appear. This is the journal version of [Kil91, KMO94].

[KMO94] E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Proc. of the 35th IEEE Symp. on Foundations of Computer Science*, pages 478–491, 1994.

[Kus89] E. Kushilevitz. Privacy and communication complexity. *SIAM J. on Discrete Mathematics*, 5(2):273–284, 1992. Preliminary version in *FOCS '89*, 1989.

[MR92] S. Micali and P. Rogaway. Secure computation. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, vol. 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer-Verlag, 1992. An updated version presented at: *Workshop on Multi-Party Secure Computation*, Weizmann Inst., Israel, June 1998.

[Nao89] M. Naor. Bit commitment using pseudorandom generators. *J. of Cryptology*, 4:151–158, 1991. Preliminary version in *Advances in Cryptology – CRYPTO '89*, 1989.

[Rab81] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[Yao82a] A. C. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 160–164, 1982.

[Yao82] A. C. Yao. Theory and application of trapdoor functions. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.