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

Amos Beimel[1], Tal Malkin[2,3], and Silvio Micali[2]

[1] Division of Engineering and Applied Sciences
Harvard University, 40 Oxford st., Cambridge, MA 02138
beimel@deas.harvard.edu
Supported by grants ONR-N00014-96-1-0550 and ARO-DAAL03-92-G0115.
[2] Laboratory for Computer Science, Massachusetts Institute of Technology
545 Technology sq., Cambridge, MA 02139
[3] tal@theory.lcs.mit.edu
Supported by DARPA grant DABT63-96-C-0018.

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, $f : S_1 \times S_2 \to S_3$ (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.

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

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 80–97, 1999.

"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.

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 Rackoff [FMR84] improved his protocol so as to be provably secure against malicious parties. Even, Goldreich, 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][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.*

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 Subsection 1.2.

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.*

An Interpretation of Our Main Theorem. Our main theorem 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? In our paper we also provide such a characterization (actually crucial to the proof of our main theorem[6]) in terms of *insecure minors*.

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.

1.2 Comparison to Previous 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

[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].

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,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.

KILIAN'S MODEL. In [Kil91] Kilian 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 function 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]

[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. 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.

Reduction Models. *Black-box* reductions (as those of [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. However, as we shall point out in Section 4.2, such reductions are not strong enough to 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 unbounded 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*.[10]

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 define:

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

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

In all the above the superscript (A, B) will sometimes be omitted when clear from the context.

We consider the 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 $\mathrm{VIEW}_B(x, \cdot, y, \cdot)$,

2.2 Secure Computation in the Unbounded Honest-but-Curious Model

Among all notions of secure computation, the one for two unbounded honest-but-curious parties is the simplest one to formalize. In this model the parties Alice and Bob are guaranteed to follow the prescribed protocol (A, B) (namely they use the right ITMs A and B), but may try to obtain as much information as they can from their own views. Intuitively, a protocol is secure in this model if the following conditions hold: (1) Bob learns the value $z = f(x, y)$, but nothing about x (not already implied by z and y), while (2) Alice learns nothing about y (and thus nothing about z not already implied by x). A formal definition follows.

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*
$$\mathrm{VIEW}_B^{A,B}(x_0, \cdot, y, r_B) = \mathrm{VIEW}_B^{A,B}(x_1, \cdot, y, r_B).^{11}$$
Bob's Privacy: $\forall x \in S_1, \forall y_0, y_1 \in S_2, \forall r_A$,
$$\mathrm{VIEW}_A^{A,B}(x, r_A, y_0, \cdot) = \mathrm{VIEW}_A^{A,B}(x, r_A, y_1, \cdot).$$

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.*[12] 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.

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

[12] 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.

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] (a definition primarily aimed at secure computation in a multi-party scenario, such as [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 when Bob is honest and for any possible malicious behavior of Alice, Bob computes the function f correctly relative to his own input and Alice's input as 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 a possibly malicious Bob, and privacy for an honest Bob against a possibly malicious Alice.

Definition 2. *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:** $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B$,
   Correctness when both Alice and Bob are honest:
   Letting $v = \mathrm{VIEW}_B^{A,B}(x, r_A, y, r_B)$, *then* $\mathrm{OUT}_B(v) = f(x, y)$.
   Correctness when only Bob is honest: *For every strategy A' there is $\mathcal{I}_{A'} : \{0,1\}^* \to S_1$ such that, letting*

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

$$\mathrm{OUT}_B(v'_B) = f(\mathcal{I}_{A'}(v'_{A'}), y).^{13}$$

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

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

[13] By the previous condition, the mapping $\mathcal{I}_A$ (i.e., for honest Alice) gives a "correct" input, which is either x itself or "equivalent" to x, in the sense that it yields the same output $f(x, y)$. Notice that for secure computation of a function f we may restrict the protocol so that Bob always ouputs a value that is compatable with his input. That is, on input y Bob outputs a value z such that there is some x for which $f(x, y) = z$ (indeed, Bob before outputing z can always check for compatibility and output $f(0, y)$ otherwise). When restricted to these secure computation the correctness for honest Bob and Alice implies the correctness when only Bob is honest, that is, the function $\mathcal{I}_{A'}$ is guaranteed to exist.

then

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

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

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

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, it securely computes f in the unbounded honest-but-curious model.*

Definition 3 (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][14]).

Definition 4 (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,*[15]

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

Examples. As immediately apparent from their tables, each of the AND and OT functions contain an insecure minor (and actually an embedded or):

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

OT :	$(0,0)$	$(0,1)$
0	0	0
1	0	1

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

[14] 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.

[15] 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.

Proof. We break the proof into two parts; Theorem 1 follows from the following Claim 1 and Claim 2.

First, we assume that f does not contain an insecure minor and prove that f is trivial by constructing a protocol (A, B) that securely computes f against malicious unbounded parties. Fix any $x_0 \in S_1$ and $y_0 \in S_2$. The protocol (A, B), described in Fig. 1, has a single round of communication (one message sent from A to B), and is deterministic (namely A and B ignore their random inputs).

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

Proof. We prove our claim by showing that f is securely computed against unbounded malicious parties by the following protocol (A, B) described in Fig. 1.

Protocol (A, B)

A, on input $x \in S_1$:
send to Bob the message $a \stackrel{\text{def}}{=} f(x, y_0)$.
B, on input $y \in S_2$, upon receipt of the message a from Alice:
Find the lexicographically first x_1 such that $f(x_1, y_0) = a$
Set the output $\text{OUT}_B(y, r_B, a)$ to $f(x_1, y)$.
If no such x_1 exists, set the output $\text{OUT}_B(y, r_B, a)$ to $f(x_0, y)$.

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

We first prove the correctness of the protocol. Recall that x and y are the inputs held by honest Alice and honest Bob respectively. Correctness when both parties are honest follows since for any message $a = f(x, y_0)$ sent by honest Alice, an honest Bob finds x_1 such that $f(x, y_0) = f(x_1, y_0)$. Since f does not contain an insecure minor then it must hold that $f(x, y) = f(x_1, y)$ (otherwise x, x_1, y_0, y constitute an insecure minor). Thus, Bob's output – $f(x_1, y)$ – is correct.

To prove correctness when only Bob is honest, we first define the following input function $\mathcal{I}_{A'} : \{0,1\}^* \to S_3$ where if there is no x_1 such that $a = f(x_1, y_0)$ then $\mathcal{I}_{A'}(x, r_A, a) = x_0$ and otherwise $\mathcal{I}_{A'}(x, r_A, a)$ is the lexicographically first x_1 such that $a = f(x_1, y_0)$. Notice that the input function $\mathcal{I}_{A'}$ is the same for every adversary A'. By the definition of $\mathcal{I}_{A'}$ it always holds that $\text{OUT}_B(y, r_B, a) = f(\mathcal{I}_{A'}(x, r_A, a), y)$ and correctness follows.

Alice's privacy follows by observing that all information sent to Bob, namely $f(x, y_0)$, can be computed by Bob alone from the output of the function $f(x, y)$. This is because Bob can find some x' such that $f(x', y) = f(x, y)$, and conclude that $f(x, y_0) = f(x', y_0)$ (since f does not contain an insecure minor). Bob's privacy follows immediately from the fact that this is a one-round protocol, where Bob sends no information to Alice, and thus her view is clearly identical for any input he 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 there is a 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. We assume, for sake of contradiction, that f contains an insecure minor.

Let x_0, x_1, y_0, y_1 constitute an insecure minor of f, that is 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,

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

for every r_A. By ranging over all possible r_A we get

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

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

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

for every 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,

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

We next use the following proposition, proved by [CK89] to hold for every protocol, to argue that this transcript is equally distributed even if we fix the random input of Bob.

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

In other words, if changing the inputs for both Alice and Bob yields the same transcript, then changing the input for Alice only (or Bob only) will also yield the same transcript.

Fix arbitrary random inputs q_A and q_B, and let $t = \text{TRANS}(x_0, q_A, y_1, q_B)$. By Equation (1), there exist q'_A, q'_B such that $\text{TRANS}(x_1, q'_A, y_1, q'_B) = t$, which by Proposition 1 implies that $t = \text{TRANS}(x_1, q'_A, y_1, q_B)$.

Now, by Proposition 1, it holds that

$$\Pr_{r_A,r_B}[\text{TRANS}(x_0,\cdot,y_1,\cdot)=t\,] = \Pr_{r_A}[\text{TRANS}(x_0,\cdot,y_1,q_B)=t\,]\cdot\Pr_{r_B}[\text{TRANS}(x_0,q_A,y_1,\cdot)=t\,], \quad (2)$$

and similarly

$$\Pr_{r_A,r_B}[\text{TRANS}(x_1,\cdot,y_1,\cdot)=t\,] = \Pr_{r_A}[\text{TRANS}(x_1,\cdot,y_1,q_B)=t\,]\cdot\Pr_{r_B}[\text{TRANS}(x_1,q'_A,y_1,\cdot)=t\,]. \quad (3)$$

By Proposition 1, for every r_B

$$\text{TRANS}(x_0,q_A,y_1,r_B)=t \text{ if and only if } \text{TRANS}(x_1,q'_A,y_1,r_B)=t$$

Hence,

$$\Pr_{r_B}[\text{TRANS}(x_0,q_A,y_1,\cdot)=t\,] = \Pr_{r_B}[\text{TRANS}(x_1,q'_A,y_1,\cdot)=t\,]. \quad (4)$$

Since by (1) $\Pr_{r_A,r_B}[\text{TRANS}(x_0,\cdot,y_1,\cdot)=t\,] = \Pr_{r_A,r_B}\left[\text{TRANS}(x_1,\cdot,y_1,\cdot)=t\,\right]$, and from (2),(3),(4), we get that, for every q_B and t,

$$\Pr_{r_A}[\text{TRANS}(x_0,\cdot,y_1,q_B)=t] = \Pr_{r_A}[\text{TRANS}(x_1,\cdot,y_1,q_B)=t].$$

That is,

$$\text{TRANS}(x_0,\cdot,y_1,q_B)=\text{TRANS}(x_1,\cdot,y_1,q_B). \quad (5)$$

for every q_B.

Recall that the view of Bob is defined as his input, his random input, and the transcript of the communication. By Equation (5), for every q_B the communication transcript between Alice and Bob is identically distributed when the inputs of Alice and Bob are x_0, y_1 and when their inputs are x_1, y_1. In both cases Bob has the same input y_1 and random input q_B, so Bob's view is identically distributed in both cases, namely for every q_B it holds that

$$\text{VIEW}_B(x_0,\cdot,y_1,q_B)=\text{VIEW}_B(x_1,\cdot,y_1,q_B). \quad (6)$$

Equation (6) contradicts the correctness requirement from 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. Thus, we have reached a contradiction, which concludes the proof of the claim. □

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 of Theorem 1 (rather than from its statement). That proof, shows that, if a function f is computable in the unbounded two-party malicious model, then it is so computed by the protocol of Fig. 1, in which only a single message is exchanged (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 function that contains an insecure minor is *complete* for secure computation. That is, every non-trivial function is complete.

We shall consider secure computation in the (computationally) bounded malicious model. That is, the computation is secure provided that the (malicious) parties run in polynomial time. Thus, for the privacy conditions to hold, the appropriate probability distributions are only required to be *indistinguishable* by polynomial time Turing Machines (rather than identical as in the unbounded case of Definition 2). In our proof we also consider the bounded honest-but-curious model. For lack of space we do not give precise definitions of secure computation in the bounded models, definitions which are much more involved and complex than the definitions in the unbounded models. Such definitions can be found, e.g., in [Gol98]. We note that our results hold for all reasonable definitions of secure computation in these model.

4.1 Reductions and Completeness

As usual, the definition of completeness relies on that of a *reduction*.

Definition 5 (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*[16] *from*

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

any protocol securely computing f in the bounded malicious model (respectively, in the bounded honest-but-curious model) to a protocol securely computing g in the same model.

Definition 6 (Completeness). *The 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).*

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. (Even more informally, f is "harder" to compute securely than g.)

4.2 Our Reduction vs. Black-Box Reductions

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, the all-or-nothing nature of two-party secure computation. For instance, the (non-trivial) OR function is not complete under black-box reductions [Kil99][17]. Hence, black-box reductions do not give any indication regarding which is the minimal assumption necessary for implementing OR securely. On the other hand, using our notions of reductions and completeness, our main theorem implies that the complexity assumptions necessary for implementing OR securely are exactly the same as for all other non-trivial functions.

Let us now emphasize that our reductions and completeness satisfy basic expected properties of such notions.

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 computing f securely in the bounded malicious model is also sufficient for securely computing g in the same model.*

Proof. Consider a protocol (A_f, B_f) that securely computes f under some assumption ASSUM_f. Since g reduces to f, we can apply the transformation from (A_f, B_f) to obtain a protocol (A_g, B_g) such that if (A_f, B_f) securely computes f then (A_g, B_g) securely computes g. Thus, if ASSUM_f holds then (A_g, B_g) securely computes g. ■

Furthermore, by definition, these reductions are transitive:

[17] Indeed, it is not hard to see that 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 prevented, any protocol which is unconditionally secure using an OR black-box can be transformed into an unconditionally secure protocol, implying that only trivial functions can be black-box reduced to OR.

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.*

Proof. Consider a protocol (A_g, B_g) that computes g securely. The transformation from any protocol (A_f, B_f) securely computing f to a protocol securely computing g ignores (A_f, B_f) and outputs (A_g, B_g). ■

Remark 1. We stress that our notion of completeness highlights the all-or-nothing nature of secure computation. Furthermore, 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 Outline. Although we aim towards the bounded malicious model, our proof of Theorem 2 wanders through the bounded honest-but-curious model (more direct proofs seem problematic[18]). We first prove that every non-trivial function is complete in the honest-but-curious model. We then use standard techniques of [GMW87] to transform any secure protocol in the bounded honest-but-curious model into a secure protocol in the bounded malicious model. In general this transformation requires some complexity assumptions, however in our case the protocol in the honest-but-curious model implies these assumptions. Thus, combining the above steps, every non-trivial function is complete in the malicious model.

Proof. We start by proving the analogue of Theorem 2 for the honest-but-curious model.

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

Proof. 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.

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 $f(x_1, y_1) = c$.

Protocol $(A_{\text{OT}}, B_{\text{OT}})$

A_{OT}'s **input:** $\beta_0, \beta_1 \in \{0,1\}$
B_{OT}'s **input:** $\imath \in \{0,1\}$
A_{OT}'s and B_{OT}'s **code:** Execute protocol (A_f, B_f) on input $x_{\beta_0}, y_{\overline{\imath}}$.
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_{OT}'s **output:** If $z_\imath = b$ then output 0, else output 1.

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

In Fig. 2 we describe a protocol $(A_{\text{OT}}, B_{\text{OT}})$ which securely computes OT using this insecure minor and the protocol (A_f, B_f).

In Protocol $(A_{\text{OT}}, B_{\text{OT}})$ it holds that $z_\imath = f(x_{\beta_\imath}, y_1)$, and, thus, $z_\imath = b$ if $\beta_\imath = 0$ (and $z_\imath = c \neq b$ otherwise), implying 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 $(A_{\text{OT}}, B_{\text{OT}})$ 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 B_{OT} should not learn any information about $\beta_{\overline{\imath}}$. However, the only information that A_{OT} sends that depends on $\beta_{\overline{\imath}}$ are during the execution of (A_f, B_f) on input $(x_{\beta_{\overline{\imath}}}, y_0)$ and, thus, $z_{\overline{\imath}} = a$ for both values of $\beta_{\overline{\imath}}$. By the fact that (A_f, B_f) computes f securely, B_f does not learn any information on $\beta_{\overline{\imath}}$. □

Note that the above protocol is secure only if B_{OT} is honest. Also note that in protocol $(A_{\text{OT}}, B_{\text{OT}})$ it is important that only B_f gets the outputs z_0 and z_1 of (A_f, B_f). That is, if A_{OT} gets z_0 or z_1 then she can learn B_{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).

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 Sketch. Let g be any finite function. We need to prove that g reduces to f in the bounded malicious model. We are promised that g reduces to f in the bounded honest-but-curious model. That is, there is a transformation from any protocol securely computing f to one securely computing g in the bounded honest-but-curious model.

To obtain a protocol securely computing g in the malicious model, we proceed as follows. First, there exists a transformation mapping any protocol that

[18] For example, we cannot use Kilian's reduction [Kil91] from OT to a *two-sided* computation of OR in the bounded malicious model.

securely computes OT in the bounded honest-but-curious model into a one-way function [IL89]. Second, since f is complete in the bounded honest-but-curious model, this implies that there exists a transformation mapping any protocol that securely computes f in the bounded honest-but-curious model into a one-way function. Third, one-way functions imply pseudo-random generators [HILL91], which in turn imply bit commitment [Nao89]. Finally, bit commitment implies that it is possible to transform any protocol securely computing an arbitrary function g in the bounded honest-but-curious model into a protocol securely computing g in the bounded malicious model [GMW87]. Putting the above together, we obtain a transformation from a protocol securely computing f in the bounded honest-but-curious model to one computing the function g in the bounded malicious model. □

We are ready to complete the proof of Theorem 2. By Theorem 1 any non-trivial function f contains an insecure minor. Thus, by Claim 3 and Claim 4, f is complete in the bounded malicious model. ■

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.

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

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. Prelim. version in *STOC '89*, 1989.

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

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.

GMR85. S. Goldwasser, M. 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. 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 Comp.*, 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 *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 1988.

HILL91. J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report TR-91-068, International Computer Science Institute. 1991.

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.

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. Personal communication. 1999.

KKMO98. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. 1998. To appear in *SIAM J. on Computing*. 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 *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.

Yao82. A. C. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 160–164, 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.