Secure Computation

(Abstract)

Silvio Micali* Phillip Rogaway†

Abstract

We define what it means for a network of communicating players to *securely* compute a function of privately held inputs. Intuitively, we wish to *correctly* compute its value in a manner which protects the *privacy* of each player's contribution, even though a powerful *adversary* may endeavor to disrupt this enterprise.

This highly general and desirable goal has been around a long time, inspiring a large body protocols, definitions, and ideas, starting with Yao [1982, 1986] and Goldreich, Micali and Wigderson [1987]. But all the while, it had resisted a full and satisfactory formulation.

Our definition is built on several new ideas. Among them:

- Closely mimicking an *ideal evaluation*. A *secure* protocol must mimic this abstraction in a *run-by-run* manner, our definition depending as much on individual executions as on global properties of ensembles.
- *Blending* privacy and correctness in a novel way, using a *special type of simulator* designed for the purpose.
- Requiring *adversarial awareness*—capturing the idea that the adversary should know, in a very strong sense, certain information associated to the execution of a protocol.

Among the noteworthy and desirable properties of our definition is the *reducibility* of secure protocols, which we believe to be a cornerstone in a mature theory of secure computation.

Invocation

The last decade has witnessed the rise of *secure computation* as a new and exciting mathematical subject. This is the study of communication protocols allowing several parties to perform a correct computation on some inputs that are and should be kept private. As a simple example, the parties want to compute the tally of some privately held votes. This new discipline is extremely subtle, involving in novel ways fundamental concepts such as probabilism, information, and complexity theory.

In the making of a new science, finding the *right* definitions can be one of the most difficult tasks: from relatively few examples, one should handle cases that have not yet arisen and reach the highest possible level of generality. It is the purpose of this work both to identify the right notion of secure computation and prove the right fundamental properties about it.

In the last few years, cryptography has been very successful in identifying its basic objectives, properly defining them, and successfully solving them. Secure encryption, secure pseudorandom generation, secure digital signatures, and zero-knowledge proofs—concepts that appeared forever elusive—have all found successful formalizations and solutions. But in contrast to these successes,

*MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

†IBM, 11400 Burnet Road, Austin, TX 78758. Work performed in part while the author was at the MIT Laboratory for Computer Science, and in part while at Dartmouth College.

J. Feigenbaum (Ed.): Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 392-404, 1992.

and despite many beautiful and fundamental ideas that preceded us, not even a satisfactory *definition* of a secure protocol has been proposed so far. This is not surprising, since protocols are extremely complex objects: after all, by defining security for encryption, signatures, and pseudorandom generation, one is defining properties of algorithms; but to properly define protocol security, one needs instead to define properties of the *interaction* of several algorithms, some of which may be deliberately designed to *disrupt* the joint computation in clever ways. The intricacy of this scenario has often encouraged researchers to work either with definitions of security tailored to the problem at hand; or to consider broad definitions, but restricted to specific computational tasks; or to work with only intuitive notions in mind.

Lack of universally accepted definitions can only create confusion and mistakes, and it is only by reaching an *exact* understanding of what we can expect from a secure protocol that can we safely rely on them and further develop them. Powerful computer networks are already in place and the possibility of using them for new and wonderful tasks will largely depend on how successful this development will be.

In These Proceedings

A full description of our notion of secure computation is beyond the scope of a proceedings' abstract. For this article, we have revised the introduction to [MR91], incorporating a *very* high-level description of the definition and a brief comparison with other notions which have been offered in the literature.

Secure-Computation Problems

What is secure computation about? Informally, it consists of finding a communication *protocol* that allows a group of *players* to accomplish a special type of *task*, despite the fact that some of them may try to sabotage this enterprise. This said, we now explain terms. Let's start with the easy ones.

Players (also called *processors* or *parties* for variation of discourse) can be thought as people, each possessing a personal computer, and capable of exchanging messages. A *protocol* is a set of instructions for the players to follow for sending these messages. The rules of the game are as follows: (1) in executing a protocol, some of the participants may be *bad*, thereby disregarding their instructions and cooperating to disrupt the joint effort; (2) no trusted device or external entity is available; (3) every good party can perform private computation (i.e., computation unmonitored by the bad players).

What is a secure protocol supposed to accomplish? We start by looking at a few archetypal examples. Since our aim is to exemplify various issues and key desiderata that may inspire us to properly define secure computation, in the following list we credit the one who first *posed* the problem.

1. THE MILLIONAIRES PROBLEM (Yao, [Ya82a]). Two millionaires wish to find out who is richer, though neither is willing to reveal the extent of his fortune. Can they carry out a conversation which identifies the richer millionaire, but doesn't divulge additional information about either's wealth?

2. THE DIGITAL VOTING PROBLEM (Chaum, [Ch81]). Is it possible for a group of computer users to hold a secret-ballot election on a computer network?

3. THE INDEPENDENT ANNOUNCEMENT PROBLEM (Chor, Goldwasser, Micali, and Awerbuch [CGMA85]). A group of players want to exchange messages so as to announce their secret

values independently. That is, what the bad players announce cannot be chosen based on the values of the good players.

4. THE COIN FLIPPING PROBLEM (Blum, [Bl82]). How can Alice and Bob, speaking to one another over the telephone, agree on a random, unbiased coin flip—even if one of them cheats to try to produce a coin flip of a certain outcome?

5. THE OBLIVIOUS TRANSFER PROBLEM (Rabin, [Ra81]). Is it possible for Alice to send to Bob a message m in such a way that (i) half the time, Bob gets m; (ii) the other half of the time, Bob gets nothing; and (iii) Alice never knows which of the two events has occurred?

6. THE MENTAL POKER PROBLEM (Shamir, Rivest and Adleman, [SRA81]). Can a group of players properly shuffle and deal a deck of cards over the phone?

Privacy and Correctness

Even the above short list illustrates the enormous variety of types of goals for secure protocols. There may be two parties or many. The output of a protocol may be a single value known to all players (as in digital voting), or to only one of them (as in an oblivious transfer), or it may be a private value for each player (as in mental poker). The output may depend on the players' initial state deterministically (as in the first three problems), or probabilistically (as in the last three problems).

What do such heterogeneous problems have in common, then? Essentially, that the joint computation should both be *private* and *correct*: while preserving the privacy of individually held data, the joint computation manages to correctly perform some computational task based on this data. Correctness and privacy may seem to be conflicting requirements, and capturing in the most general sense what simultaneously meeting them means (within our rules of the game) is quite difficult. As we explain in the full paper, to obtain a satisfactory notion of security privacy and correctness should not be handled independently (like in all prior work), but need to be *blended* in the proper way.

Prior and Related Definitions

Y-EVALUATION. Distilling a common thread in many prior examples of secure computation, Yao proposed the following general problem [Ya82a]. Assume we have n parties, $1,\ldots,n$. Each party i has a *private* input, x_i, known only to him. The parties want to compute a given function f on their own inputs while maintaining the privacy of these inputs. In other words, they want to compute $y = f(x_1,\ldots,x_n)$ without revealing to any player more about the private inputs than the output itself implicitly reveals. If the function is vector-valued, $\vec{y} = f(x_1,\ldots,x_n)$, where $\vec{y}$ has n components, it is desired that every party i privately learn the i-th component of $\vec{y}$.

Yao also proposed a notion for what it means for a protocol to solve the above problem. Roughly said, his formalization attempts to capture the idea that the worst the bad players can do to disrupt a computation is to choose alternative inputs for themselves, or quit in the middle of the computation. We will refer to this notion of security as *Y-evaluation*. Subsequently [Ya86], Yao strengthened his notion of a Y-evaluation so as to incorporate some *fairness* constraint. A fair protocol is one in which there is very little advantage to be gained by quitting in the middle. That is, the protocol takes care that, at each point during the execution, the "informational gap" among the players is small. The study of fair protocols was started earlier by Luby, Micali and Rackoff [LMR83], and progressed with the contributions of [Ya86, BG89, GL90].

GMW-GAMES. A more general notion for security has been introduced by Goldreich, Micali and Wigderson [GMW87]. They consider secure protocols as implementations of abstract, but computable, games of partial information. Informally, ingredients of such an n-player game are an arbitrary set of *states*, a set of *moves* (functions from states to states), a set of *knowledge functions* (defined on the states), and a vector-valued *outcome function* (defined on the states) whose range values have as many components as there are players. The players wish to start the game by probabilistically selecting an initial global state, unknown to everyone. Then the players take turns making moves. When it is the turn of player i, a portion $K(S)$ of the current global state S must be privately revealed to him; here K denotes the proper knowledge function for this stage of the game. Based on this private information, player i secretly selects a move μ, thereby the new, secret, state must become $\mu(S)$. At the end of the game, each player privately learns his own component of the outcome function evaluated at the final state.

[GMW87] envisioned a notion of security which would mimic the abstract game in a "virtual" manner: states are virtually selected, moves act on virtual states, and so on. We will refer to their notion as *GMW-games*. Again putting aside how this can be achieved, let us point out an additional aspect of their notion: namely, in a GMW-game, bad players cannot disrupt the computation *at all* by quitting early. (This condition can indeed be enforced, in a certain communication model, whenever the majority of the players are honest)

OTHER PRIOR WORK. Several noteworthy variants of Y-evaluation and GMW-games have been proposed, with varying degrees of explicitness and care. These definitional ideas include, most notably, the work of Galil, Haber and Yung [GHY87], Chaum, Damgård and van de Graff [CDG87], Ben-Or, Goldwasser and Wigderson [BGW88], Chaum, Crépeau and Damgård [CCD88], Kilian [Ki89], and Crépeau [Cr90].

CONCURRENT WORK. Early in our effort we told our initial ideas (like merging privacy and correctness) to Beaver, who later pursued his own ones in [Be91a, Be91b].

Later, we collaborated with Kilian in developing definitions for secure function evaluation. This collaboration was enjoyable and profitable, and its fruits are described in [KMR90].

Concurrently with the effort of [KMR90], Goldwasser and Levin [GL90] independently proposed an interesting approach to defining secure function evaluation.

Critique of These Definitions

Definitions cannot, of course, be "wrong" in an absolute sense; but we feel that all previous ones were either vague, or not sufficiently general, or considered "secure" protocols that should have not been called such at a closer analysis. We thus cheerfully decided to clarify the intuitive notion of secure computation. Little we knew that we had taken up a two-year commitment!

Let us very briefly critique some of the mentioned work.

Y-EVALUATION. Though Yao should be credited for presenting his notion with great detail[1], in our opinion his ideas do not fully capture the fundamental intuition of secure computation, leading to several difficulties.

Blind input correlation. One of these difficulties we call "blind input correlation." Namely, a two-party Y-evaluation, while preventing a bad player from directly learning the good player's input, might allow him to choose his own input so to be correlated with the good player's. For example, it

[1] This precision, however, was made heavier from lacking more modern constructs for discussing these issues, like the notion of a simulator developed by Goldwasser, Micali and Rackoff [GMR89].

is not ruled out that, whenever the good player starts with secret input x, the bad player, without finding out what x was, can force the output to be, say, $y = f(x,x)$ (or $y = f(x,-x)$, or ...). In some context, this correlation may result in (what would be considered by most people) a loss of security.[2]

Privacy/correctness separation. Y-evaluation considers privacy and correctness as separate constraints and, though with a different terminology, considers secure a protocol that is *both* private and correct. Indeed, privacy and correctness are the fundamental aspects of secure computation, but, as we describe in the full paper, the logical connective "and" *does not* combine them adequately together. In particular, there exists functions f and protocols P such that, under most reasonable-sounding definitions (including Yao's), P correctly computes f; and P is private leaking only f; and yet P is definitely not a good protocol to securely compute f. Basically, the problem is that there can be *tradeoffs* between privacy and correctness, and simply demanding their conjunct does not respect this possible interplay.

GMW-GAMES. GMW-games are most general and powerful, and they are endowed with very strong intuitive appeal. However, the author's main goal was to propose the first general solution to the problem of securely evaluating a function, and so they were less concerned with arriving at a fully general, "protocol-independent" notion of security. In particular, the authors notion of security was tailored for protocols that begin with a "commit" stage, and then perform computation on these committed inputs. While their algorithmic structure has proved to be very successful (indeed, all subsequent protocols designed for the task share it), it should not be embedded in the general definition of our goal. Finally, the authors did not provide full help in their proceedings paper for turning the intuition described into a successful formalization, and several wrong choices could still be made from the level their definition was left at.

CONCURRENT WORK. Concurrent work does not suffer from this drawback, but has other shortcomings. Beaver [Be91a] does not blend the various goals for a secure protocol, but treats them as separate requirements, as Yao first did and incurring in the same type of difficulties. In [Be91b], the author avoids this problem, but offers a notion of security weaker than ours (once cast in the same framework), and one which calls "secure" some protocols which we feel should not be considered as such. Security in the sense of Goldwasser and Levin, in turn, seems to be weaker than the simulation-based notion of [Be91b]. Both, then, admit pitfalls including the following:

Privacy/correctness unification. If Yao goes to one extreme in separating privacy and correctness, [GL90, Be91b] go to the other, completely merging them. We do not have space to adequately describe their definitions, but the net result is that if privacy is to be achieved in a computational sense (which, for example is necessarily the case when everyone communicates on a broadcast channel, using encryption to hide private inputs), then correctness also is achieved in a computational sense. While it is OK to say that the adversary cannot understand someone's input because she does not have the resources to make sense of it, it is less acceptable to output wrong results and just be saved by the fact that they "look right" with respect to time-bounded computation. Consider, for instance, the computation that consists of outputting a random quadratic reside modulo

[2] Let's see what its effects might be on some of the discussed secure-computation problems. Consider solving the digital voting problem by Y-evaluating the tally function. Then the bad players —though ignoring the electoral intentions of a given good player— might succeed in voting as a block for the opposite candidate. Should we consider this a secure election? We believe not. Similarly, trying to solve the independent announcement problem by Y-evaluating the "concatenation" function (i.e., the function that, on inputs $x_1, \ldots, x_n$, returns the single value $x_1 \# \ldots \# x_n$), a bad player might always succeed in announcing the same value as a given good player. Indeed, a poor case of independence!

a composite number whose factorization is unknown by the players. Then a secure protocol in the sense of [GL90] or [Be91b] may well consist of outputting a random quadratic *non*residue of Jacobi symbol +1. Under the proper complexity assumption, no one can notice the difference—yet it seems (to us) that this protocol is not a correct one for performing the specified task.

Input absence. Sometimes we define a function part of the purpose of which is to check that its inputs fall in some relation to one another: for simplicity, consider $f(\vec{x}) = 1$ if $x \in OK$, $f(\vec{x}) = 0$ otherwise. A secure computation of f should, when it evaluates to 1, convince the participants that they collectively hold an $\vec{x} \in OK$. However, protocols secure in the sense of [GL90] or [Be91b] allow the good players to each output a 1 even though the bad guys have never chosen their contribution $\vec{x}_T$—and, in fact, even when there is *no* such contribution which would result in the function evaluating to 1!

Reducibility? In light of the difficulty of designing secure protocols, it is essential that the definition of security supports a "reducibility property" which ensures the possibility of constructing complex secure protocols by properly combining simpler secure protocols in a black-box manner. (This will be further described shortly.) Unfortunately, we do not see any way of proving the reducibility property we are after from their alternative definitions.

Our Definitions

Our notion of *secure computation* solves the above difficulties, and other ones as well. We plan to build up our definition in two stages. First, as our alternative to Y-evaluation, we define *secure function evaluation*, to which this paper is devoted. Our notion is quite powerful and expressive; for instance, the first three secure-computation problems described earlier are straightforwardly solvable by securely evaluating the proper function. After developing secure function evaluation, we hint how this notion can be successfully extended to that of a *secure game*, our way of fully specifying GMW-games. (The full paper is already quite long, and a different one should be devoted to a detailed treatment of this extension. Also, the present treatment restricts its definitions to there being three or more players.) Secure games capture, in our opinion, the very notion of secure computation. Quite reassuringly, all six problems specified earlier are straightforwardly solved by "playing" a secure game.

The basic intuition behind secure function evaluation is the same one put forward, in quite a different language, by [GMW87]. In essence, two scenarios are considered: an *ideal* one in which there is a trusted party helping in the function evaluation, and a *realistic* one in which the trusted party is simulated by running a protocol. Security is a property of a realistic evaluation, and consists of achieving "indistinguishability" from the corresponding ideal evaluation. While remaining quite informal, let us at least be less succinct.

IDEAL FUNCTION EVALUATION. In an ideal function evaluation of a vector-valued function f there is an external *trusted party* to whom the participants privately give their respective secret inputs. The trusted entity will not divulge the received inputs; rather, it will correctly evaluate function f on them, and will privately hand component i of the result to party i. An *adversary* can interfere with this evaluation as follows. At the very beginning, before any party has given his own input to the trusted party, she can corrupt a first player and learn his private input. After this, and possibly based on what she has just learned, the adversary may corrupt a second player and learn this input, too. This continues until the adversary is satisfied. At this point, the adversary chooses alternative, *fake* inputs for the corrupted players, and all parties give the their inputs to the trusted authority—the uncorrupted players giving their initial inputs, and the corrupted players giving their new, fake inputs. When the proper, individual outputs have been returned by the trusted party,

the adversary learns the value of the output of every corrupted player. The adversary can still corrupt, one by one, additional players, learning both their inputs and outputs when she does so.

It should be noticed that in such an ideal evaluation the adversary not only learns the inputs of the players she corrupts, but by choosing properly their substitutes, she may learn from the function's output value quite a bit about the other inputs as well.

IDEAL VS. SECURE FUNCTION EVALUATION. While the notion of an ideal function evaluation has been essentially defined above, formalizing the notion of a secure function evaluation is much more complex; we will do it in the next few sections. Here let us just give its basic intuition. To begin with, there no longer is any trusted party; the players will instead try to simulate one by means of a protocol. The adversary can still corrupt players, but this time a corruption will be much more "rewarding." For not only will she learn the private inputs of corrupted players, but also their current computational state in the protocol (and a bit of additional information as well). She will receive all future message addressed to them and she will get control of what messages they are responsible for sending out.

Given the greater power of the adversary in this new setting, it is intuitively clear that a protocol for function evaluation cannot perform "better" than an ideal function evaluation; but it can do much worse! Roughly said, a secure function evaluation consists of simulating *every important aspect* of an ideal function evaluation, to the *maximum extent possible*, so that a secure protocol does *not* perform "significantly worse" than the corresponding ideal protocol.

Setting the stage of security in terms of the above indistinguishability is an important insight of the [GMW87] work. The main difference with their work, though, and the real difficulty of ours, is not so much in the spirit of the solution as in properly realizing what "maximum possible" should mean, and identifying what are the important features, implicit (and perhaps hidden) in an ideal function evaluation that should (and can!) be mimicked in a secure function evaluation.

We point out that none of the alternative definitions proposed for secure function evaluation is motivated by attempting to mimic the ideal evaluation of a function in so direct and strong a manner. Rather, they intend that a protocol should be called "secure" if and only if it is, intuitively, a secure computation. We, instead, are willing to call "secure" only those protocols which mimic the ideal evaluation extremely closely—even if this means excluding "intuitively correct" protocols from being called "secure."

Key Features of Our Definitions

Let us now highlight the key features of our definitions. These we distinguish as *choices* and *properties*. The former are key technical ideas of our notion of security. The latter are key desiderata: each one is a *condition-sine-qua-non* for calling a protocol secure. Notice, though, that we do not force these necessary conditions into our definitions in an artificial manner; rather, we derive them naturally as consequences (hence the name "properties") of our notion of security, and thus of our technical choices.

Key Choices

BLENDING PRIVACY AND CORRECTNESS. Secure protocols are more than just correct and private. Simply requiring simultaneous meeting of these two requirements leads to several "embarrassing" situations. In a secure protocol, correctness and privacy are *blended* a deeper manner. In particular, privacy —a meaningful notion all by itself— is taken to mean that the protocol admits a certain type of *simulator*, and correctness is a concept which we define *through the same simulator* proving the

protocol private. This merging of privacy and correctness avoids calling secure protocols those which clearly are not, and is a main contribution of this paper, as well as one of our first achievements in the course of this research. We are pleased to see that our idea has been adopted by other researchers in the area.

ADVERSARIAL AWARENESS. In the ideal evaluation of f, not only is there a notion of what inputs the adversary has substituted for the original ones of the corrupted players, but also that she is *aware* of what these substituted inputs are! Similarly, she is *aware* of the outputs handed by the trusted party to the corrupted players. We realize that this is a crucial aspect of ideal evaluation, and thus one that should be preserved as closely as possible by secure evaluation. Indeed, *adversarial awareness* is an essential ingredient in obtaining the crucial *reducibility* property discussed below.

TIGHT MIMICRY. Our definition of a secure protocol mimics the ideal evaluation of a function f in a very "tight" manner. (For those familiar with the earlier definition of zero knowledge, the definition of security relies less on "global" properties of ensembles of executions and more on *individual* executions.) In each particular run of a secure protocol for evaluating f, one may "put a finger on" *what inputs* the adversary has effectively substituted for the original ones of the corrupted players; *when* in the computation this has happened; *what* the adversary and the players get back from the joint computation, and *when* this happens; and these values are guaranteed (almost certainly) to be exactly what they should be — based on f, the inputs the adversary has effectively substituted, and the inputs the good players originally had.

Key Properties

MODEL INDEPENDENCE. As we said a secure protocol is one that "properly" replaces the trusted external party of ideal evaluation with exchanging messages. There are, however, several possibilities for exchanging messages. For instance, each pair of participants may be linked by a secure communication channel (i.e., the adversary cannot hear messages exchanged by uncorrupted people); alternatively, each pair of players may have a dedicated, but insecure, communication channel; else, the only possible communication may consist of broadcasting messages, and so on. Though for ease of presentation we develop our notion of security with respect to a *particular*, underlying communication model, our notion of security is essentially *independent* of the underlying communication model. Indeed, we prove that the existence of secure protocol in one model of communication entails their existence in all other "reasonable" models. This proof is highly constructive: we show that, for any two rich-enough communication models, there exists a "compiler" that, given a protocol secure in the first model, generates a protocol for the same task which is secure in the second. (Interestingly, the proof, though often considered folklore, turned out to be quite difficult!)

SYNTACTIC INDEPENDENCE. Our definition of security is independent of the "syntax" in which a protocol is written. Designing a secure protocol is easier if one adopts a syntactic structure à la [GMW87]; that is, having the parties first execute a "committal phase" in which they pin down their inputs while keeping them still secret[3] and then they compute on these committed inputs. A different type of syntactic help consists of assuming that a primitive for securely computing some simple function is given, and then reducing to it the secure computation of more complex functions. This also simplifies the design of secure computation protocols (and actually presupposes that secure protocols enjoy the reducibility property we discuss below.) While it is alright to use a right syntax to lessen the difficulty of designing secure protocols, we insist that our definition of

[3]This secret commitment is called verifiable secret sharing, a notion introduced by [CGMA85]. For a precise definition and worked out example see [FM90]

security be independent of any specific syntactic restriction for a protocol to be called secure. (Of course, though we insist at remaining at an intuitive level here, being secure is itself an enormous restriction for a protocol, but of a different nature.)

INPUT INDEPENDENCE. There is yet a third, and crucial, type of independence property. In the ideal evaluation of a function f, the fake values the adversary chooses are completely *independent* of the values held by *uncorrupted* players. Of course, a secure protocol should have this property too, as closely as is possible. (It is a rather subtle matter how to state this goal satisfactorily.) After the proper definitions are in place, we show that our definition of security captures independence in an *extremely strong sense*—not by "adding it in" as a desired goal, but, rather, by it being a *provable property* of any secure protocol.

REDUCIBILITY. Let us describe this goal. Suppose you have designed a secure protocol for some complicated task—computing some function f, say. In an effort to make more manageable your job as protocol designer, you assumed in designing the protocol that you had some *primitive*, g, in hand. You proved that your protocol P^g for computing f was secure in a "special" model of computation — one in which an "ideal" evaluation of the primitive g was provided "for free." One would want that you obtain a secure protocol for f by inserting the code of a secure protocol for g wherever it is necessary in P^g that g be computed. This key goal for any "good" definition of security is surprisingly difficult to obtain, *particularly* if one adopts the most natural and innocent-looking notion of adversarial awareness (namely, the result of applying a fixed algorithm to the adversary's entire computational state). Reducibility has always been a key desideratum for us, and, to our knowledge, this is the first definition which provably achieves it.

The Definition, in a Nutshell

Here we describe, at the highest possible level, our definition of secure function evaluation.

Our notion owes much to the idea of a simulator, originally devised by Goldwasser, Micali and Rackoff [GMR89] in the context of interactive proofs. Recall that, in that context, a simulator which establishes zero-knowledge is a probabilistic algorithm which, knowing only "what it is entitled to" (whether or not $x \in L$) manages to produce a "fake" adversary view drawn from a distribution which closely resembles the distribution on "real" adversary views. To call a protocol secure, we, too, will demand the existence of a simulator—but our simulators are of a different sort. In fact, there are two major differences, which we now describe.

The first difference concerns specifying what information should be provided to the simulator: it is entitled to exactly the information available in the ideal evaluation. This is straightforwardly accomplished by equipping the simulator with a special type of oracle. Specifically, this oracle responds to two types of queries: first, *component queries*, which ask for the values of particular private inputs; and also a (single) *output query*, which asks the value of the function at a (partially) specified input, $\vec{x}_T$.

The second major difference is that instead of being an algorithm which produces a certain distribution on views, our simulator plays the role of the uncorrupted players. In fact, the adversary interacts with a simulator just as though she were interacting with the network. The "good" simulators (those which show that a protocol is secure) manage to interact with *any* adversary in a way which makes it indistinguishable to her whether it is the simulator or the network with whom she speaks. (This type of simulator, which we call an "on-line simulator," is not the same restriction as "black-box" simulation.)

Demanding the existence of such a simulator defines a strong notion of privacy. To capture security as a whole we employ the same simulator demanded for a privacy, as we now describe.

When the simulator asks the output query of its oracle, it computes this query—somehow. We add in a simple restriction: that the adversary *herself* could compute this query as a function $\mathcal{AI}$ of information which she undeniably has assess to (the "message traffic" between herself and the simulator, say). This function $\mathcal{AI}$—the *adversary input function*—captures something important in the ideal evaluation of a function f: that at some point in time the adversary inserts a value x'_T into the collaborative computation, and she is necessarily aware of what this value is and when she provides it.

Correctness is defined by looking at individual executions of the network in the presence of the adversary. We demand that in a correct evaluation of f, what the good players compute is (almost always) the function f evaluated at what the adversary input function indicates the adversary regards herself as having used for substituted values on this particular execution on behalf of the corrupted players, taken together with the good players' private inputs.

Actually, there is more to correctness than that. As mentioned, in the ideal evaluation of a function, not only is the adversary aware of what she contributes on behalf of corrupted players and when, but she gets an output back from these corrupted players, too! A secure protocol must mimic this. Thus the adversary must be able to compute, on behalf of the bad players, what is their share of the correctly computed function value. This is formalized by demanding the existence of an *adversary output function* $\mathcal{AO}$—again, something easily computed by the adversary—and, thinking of the output from a protocol execution being what the good players do output together with what the adversary "could" output by evaluating $\mathcal{AO}$, the notion of correctness is as before.

(We comment that extra care must be exercised in ensuring that the adversary input and output are properly co-ordinated with one another—roughly, that not only does the adversary regard herself as having inserted a certain value into the collaborative computation, and gotten a certain value back from it, but that these values are, in fact, the values the protocol has acted on.)

Other Scenarios

One last comment concerning the generality of our notions: that, though the full paper singles out just a few models of computation, one particular adversary, and one particular ideal scenario to mimic, it is the *thesis* underlying this work that the definition given here lifts to many other reasonable scenarios as well, a great many being easy to describe and potentially interesting. Essentially, the thesis is as follows: you specify the communications model, the ideal evaluation you are trying to imitate, and the adversary's capabilities, and we provide, making the right "syntactic modifications," a definition of what it means for a communications protocol to imitate the corresponding abstraction.

Acknowledgments

In distilling our notion of secure computation we have benefitted highly from the beautiful insights of those who preceded us.

This work may not have come about without the earlier notions of secure function computation set forth by Yao [Ya82a, Ya86], and by Goldreich, Wigderson, and the first author [GMW87].

A more recent, fundamental source of inspiration was provided by the work of Kilian [Ki89] and the joint work of Kilian and the authors [KMR90]. Some of the knots solved here were first identified and/or untangled there.

An equally crucial role was played by the work of Claude Crépeau and the first author [Cr90], which also provided us with a wonderful source of examples that proved crucial for refining our ideas.

We have also gained plenty of key insights from Goldwasser's, Rackoff's, and the first author's work on the earlier, related notion of a zero-knowledge proof [GMR89].

Next, our thanks for recent discussions with Michael Fischer, Leonid Levin, and Adi Shamir.

Last but not least, we would like to thank the many friends with which we exchanged ideas about secure protocols for so many years.

I (the first author) was fortunate to have had Manuel Blum, Shafi Goldwasser, Oded Goldreich, Charles Rackoff, and Michael Fischer as traveling companions in very heroic times, when secure protocols were a totally unexplored and hostile territory. A bit more recently, my very special thanks go to my (cryptography) students Paul Feldman, Claude Crépeau, Phil Rogaway, Mihir Bellare, and Rafail Ostrovsky, from which I continue learning an enormous amount.

I (the second author) happily thank Mihir Bellare and Joe Kilian for many nice discussions on protocols and cryptography.

References

[Be91a] D. BEAVER, "Formal Definitions for Secure Distributed Protocols," in *Distributed Computing and Cryptography — Proceedings of a DIMACS Workshop*, October 1989. (Paper not presented at workshop but invited to appear in proceedings.)

[Be91b] D. BEAVER, "Foundations of Secure Interactive Computing," these proceedings.

[BG89] D. BEAVER AND S. GOLDWASSER, "Multiparty Computations with Faulty Majority," *Proc. of the 30th FOCS* (1989), 468–473.

[BMR90] D. BEAVER, S. MICALI AND P. ROGAWAY, "The Round Complexity of Secure Protocols," *Proc. of the 22nd FOCS* (1990), 503-513.

[BF85] J. BENALOH (COHEN) AND M. FISCHER, "A Robust and Verifiable Cryptographically Secure Election Scheme," *Proc. of the 26th FOCS* (1985), 372–381.

[BGW88] M. BEN-OR, S. GOLDWASSER AND A. WIGDERSON, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," *Proc. of the 20th STOC* (1988), 1–10.

[Bl82] M. BLUM, *Coin Flipping by Telephone, IEEE COMPCON*, (1982) 133–137.

[BM82] M. BLUM AND S. MICALI, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM J. of Computing*, Vol. 13, No. 4, 1984, 850–864. Earlier version in *Proc. of the 23rd FOCS* (1982).

[Ch81] D. CHAUM, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Comm. of the ACM* 24 (2) February 1981, 84–88.

[BCC88] G. BRASSARD, D. CHAUM AND C. CRÉPEAU, "Minimum disclosure proofs of knowledge," *Journal of Computer and System Sciences*, Vol. 37, No. 2, October 1988, 156–189.

[CCD88] D. CHAUM, C. CRÉPEAU AND I. DAMGÅRD, "Multiparty Unconditionally Secure Protocols," *Proc. of the 20th STOC* (1988), 11–19.

[CDG87] D. CHAUM, I. DAMGÅRD AND J. VAN DE GRAFF, "Multiparty Computations Ensuring the Privacy of Each Party's Input and Correctness of the Result," CRYPTO-87 Proceedings, 87–119.

[CG89] B. CHOR AND E. KUSHILEVITZ, "A Zero-One Law for Boolean Privacy," *Proc. of the 21st STOC* (1989), 62–72.

[CGK90] B. CHOR, M. GERÉB-GRAUS AND E. KUSHILEVITZ, "Private Computations over the Integers," *Proc. of the 31st FOCS* (1990), 325–344. Earlier version by Chor and Kushilevitz, "Sharing over Infinite Domains," CRYPTO-89 Proceedings, Springer-Verlag, 299–306.

[CGMA85] B. CHOR, O. GOLDWASSER, S. MICALI AND B. AWERBUCH, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," *Proc. of the 26th FOCS* (1985), 383-95.

[Cr90] C. CRÉPEAU, "Correct and Private Reductions Among Oblivious Transfers," MIT Ph.D. Thesis, February 1990.

[DLM] R. DeMillo, N. Lynch, and M. Meritt, "Cryptographic Protocols," *Proc. of the 14th STOC* (1982) 383–400.

[DH76] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, 22(6) (November 1976). 644-654.

[Ed65] J. Edmonds, "Paths, Trees, and Flowers," *Canadian J. of Mathematics*, 17:449–467, 1965.

[FFS87] U. Feige, A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Identity," *Proc. of the 19th STOC* (1987), 210–217.

[FS90] U. Feige and A. Shamir, "Witness indistinguishability and witness hiding protocols," *Proc. of the 22nd STOC* (1990), 416–426.

[FS91] U. Feige and A. Shamir, "On Expected Polynomial Time Simulation of Zero Knowledge Protocols," in *Distributed Computing and Cryptography — Proceedings of a DIMACS Workshop*, October 1989.

[Fe88] P. Feldman, "One Can Always Assume Private Channels," unpublished manuscript (1988).

[FM90] P. Feldman and S. Micali, "An Optimal Algorithms for Synchronous Byzantine Agreement," MIT/LCS Technical Report TM-425 (June 1990). Previous version in *Proc. of the 20th STOC* (1988), 148–161.

[GHY87] Z. Galil, S. Haber and M. Yung, "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model," CRYPTO-87 Proceedings, 135–155.

[Go89] O. Goldreich, "Foundations of Cryptography – Class Notes," Spring 1989, Technion University, Haifa, Israel.

[GL90] S. Goldwasser and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority," CRYPTO-90 Proceedings, 75–84.

[GM84] S. Goldwasser and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, Vol. 28, No. 2 (1984), 270–299. Earlier version in *Proc. of the 14th STOC* (1982).

[GMR89] O. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. of Comp.*, Vol. 18, No. 1, 186–208 (February 1989). Earlier version in *Proc. of the 17th STOC* (1985), 291–305.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMW87] O. Goldreich, S. Micali and A. Wigderson, "How to Play Any Mental Game," *Proc. of the 19th STOC* (1987), 218–229.

[GV87] O. Goldreich and R. Vainish, "How to Solve any Protocol Problem—An Efficiency Improvement," CRYPTO-87 Proceedings, 76–86.

[Ha88] S. Haber, "Multi-Party Cryptographic Computation: Techniques and Applications," Columbia University Ph.D. Thesis (1988).

[HU79] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[Ki89] J. Kilian, "Uses of Randomness in Algorithms and Protocols," MIT Ph.D. Thesis, April 1989.

[KMR90] J. Kilian, S. Micali, and P. Rogaway, "The Notion of Secure Computation," manuscript, 1990.

[Le85] L. Levin, "One-Way Functions and Pseudorandom Generators," *Combinatorica*, Vol. 17, 1988, 357–363. Earlier version in *Proc. of the 17th STOC* (1985).

[LMR83] M. Luby, S. Micali and C. Rackoff, "How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically Biased Coin," *Proc of the 24th FOCS* (1983).

[Me83] M. MERITT, "Cryptographic Protocols." Georgia Institute of Technology Ph.D. Thesis, Feb. 1983.

[MRS88] S. MICALI, C. RACKOFF AND B. SLOAN, "The Notion of Security for Probabilistic Cryptosystems," *SIAM J. of Computing*, 17(2):412–26, April 1988.

[MR91] S. MICALI AND P. ROGAWAY, "Secure Computation," manuscript, August 1991.

[Or87] Y. OREN, "On the Cunning Power of Cheating Verifiers: Some Observations about Zero Knowledge Proofs," *Proc. of the 28th FOCS* (1987), 462–471.

[PSL80] M. PEASE, R. SHOSTAK AND L. LAMPORT, "Reaching Agreement in the Presence of Faults," *J. of the ACM* Vol. 27, No. 2, 1980.

[Ra81] M. RABIN, "How to Exchange Secrets by Oblivious Transfer," Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[RB89] T. RABIN AND M. BEN-OR, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority," *Proc. of the 21st STOC* (1989), 73–85.

[SRA81] A. SHAMIR, R. RIVEST, AND L. ADLEMAN, "Mental Poker," in *Mathematical Gardener*, D. D. Klarner, editor, Wadsworth International (1981) pp 37–43,

[TW87] M. TOMPA AND H. WOLL, "Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information," *Proc. of the 28th FOCS* (1987), 472–482.

[Ya82a] A. YAO, "Protocols for Secure Computation," *Proc. of the 23 FOCS* (1982), 160–164.

[Ya82b] A. YAO, "Theory and Applications of Trapdoor Functions," *Proc. of the 23 FOCS* (1982) 80–91.

[Ya86] A. YAO, "How to Generate and Exchange Secrets," *Proc. of the 27 FOCS* (1986).