Universally Composable Security: A New Paradigm for Cryptographic Protocols
(Extended Abstract)

Ran Canetti*

Abstract

We propose a new paradigm for defining security of cryptographic protocols, called **universally composable security.** *The salient property of universally composable definitions of security is that they guarantee security even when a secure protocol is composed with an arbitrary set of protocols, or more generally when the protocol is used as a component of an arbitrary system. This is an essential property for maintaining security of cryptographic protocols in complex and unpredictable environments such as the Internet. In particular, universally composable definitions guarantee security even when an unbounded number of protocol instances are executed concurrently in an adversarially controlled manner, they guarantee non-malleability with respect to arbitrary protocols, and more.*

We show how to formulate universally composable definitions of security for practically any cryptographic task. Furthermore, we demonstrate that practically any such definition can be realized using known techniques, as long as only a minority of the participants are corrupted. We then proceed to formulate universally composable definitions of a wide array of cryptographic tasks, including authenticated and secure communication, key-exchange, public-key encryption, signature, commitment, oblivious transfer, zero knowledge and more. We also make initial steps towards studying the realizability of the proposed definitions in various settings.

Keywords: cryptographic protocols, security analysis of protocols, concurrent composition.

1 Introduction

Rigorously demonstrating that a protocol "does its job securely" is an essential component of cryptographic protocol design. This requires coming up with an appropriate mathematical model for representing protocols, and then formulating, within that model, a *definition of security* that captures the requirements of the task at hand. Once such a definition is in place, we can show that a protocol "does its job securely" by demonstrating that it satisfies the definition of security in the devised mathematical model.

However, coming up with a good mathematical model for representing protocols, and even more so formulating appropriate definitions of security within the devised model, turns out to be a tricky business. The model should be rich enough to represent a large variety of realistic adversarial behaviors, and the definition should guarantee that the intuitive notion of security is captured, for any adversarial behavior under consideration. This in particular means that security should be maintained when the protocol is used as a component within a larger system.

In contrast, cryptographic primitives (or, *tasks*) were traditionally first defined as stand-alone protocol problems. This allowed for relatively concise and intuitive problem statement, as well as simple analysis of protocols. However, in many cases it turned out that the initial definitions were insufficient in more complex contexts, and especially when deploying protocols within larger systems or protocol environments. Examples include encryption (where semantic security [GM84] was later augmented with several flavors of security against chosen ciphertext attacks, e.g. [NY90, DDN00, RS91, BDPR98] and adaptive security [BH92, CFGN96]), commitment (where the original notions were augmented with some flavors of non-malleability [DDN00, DIO98, FF00] and equivocability, e.g., [BCC88, B96]), Zero-Knowledge protocols (where the original notions [GMRa89, GO94] were shown not to be closed under parallel and concurrent composition [GK88, F91, DNS98]), Key Exchange [BR93, BCK98, Sh99, CK01], Oblivious Transfer [R81, EGL85, GM00], and more.

One way to capture the security concerns that arise in some specific protocol environment or in a given application is to directly represent the given environment or application within an extended definition of security. (Such an approach was taken, for instance in the cases of concur-

*IBM T.J. Watson Research Center. Email: canetti@us.ibm.com.

Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)
0-7695-1390-5/02 $17.00 © 2002 IEEE

rent zero-knowledge and oblivious transfer [DNS98, GM00] as well as non-malleability of protocols [DDN00], where the definitions explicitly model several adversarially coordinated instances of the protocol in question.) This approach, however, results in definitions with ever-growing complexity, and is inherently limited in scope since it addresses only specific environments and concerns.

An alternative approach, taken in this work, is to use definitions that treat the protocol as stand-alone but guarantee *secure composition.* That is, here definitions of security inspect only a single copy of the protocol *in vitro.* Security in complex settings (where a protocol instance may run concurrently with many other protocol instances, on potentially related inputs and in an adversarially controlled way) is guaranteed via a general *composition theorem.* On top of simplifying the process of formulating definitions and analyzing protocols, this approach guarantees security in arbitrary protocol environments, even unpredictable ones which have not been explicitly stated.

In order to make such an approach (and in particular, such a composition theorem) meaningful, we first need to have a general framework in which to represent cryptographic protocols and the security requirements of cryptographic tasks. Indeed, several general definitions of secure protocols were developed over the years, e.g. [GL90, MR91, B91, BCG93, PW94, C00, HM00, DM00, PSW00, PW00]. Some of these definitions were shown to maintain security under natural composition operations. These definitions are obvious candidates for such a general framework. However, the composition operations considered in those works fall short of guaranteeing general secure composition of cryptographic protocols, especially in settings where security holds only for computationally bounded adversaries and numerous protocols may be running concurrently in an adversarially coordinated way. Moreover, many of these works choose to concentrate on the task of *secure function evaluation* which, in spite of its generality, does not capture the requirements of many cryptographic primitives, which are reactive in nature. (Secure function evaluation is the task where a set of parties wish to jointly compute a known function of their secret inputs.) We further elaborate on some of these works and their relation to the present one in Section 1.4.

This work proposes a new framework for representing and analyzing cryptographic protocols. Within this framework, we propose a general methodology for expressing the security requirements of practically any cryptographic task in a clear, concise and intuitively satisfying way. The salient property of definitions of security generated using this methodology is that they guarantee security even when the given protocol is running in an arbitrary and unknown multi-party environment. In particular, security is preserved under a very general composition operation that captures, as special cases, the standard notions of concurrent composition (with arbitrarily many instances of either the same protocol or other protocols), non-malleability, and more. We call this composition operation **universal composition,** and say that definitions of security in this framework are **universally composable (UC).**

UC definitions of security tend to be more stringent than other definitions of security. Nonetheless, we show that in settings where parties have access to a set of servers, at most a minority of which may be corrupted, standard cryptographic techniques (e.g., [BGW88, RB89, CFGN96]) can be used to carry out practically *any cryptographic task* in a universally composable way. We also formulate UC definitions of a number of well known cryptographic tasks, such as authenticated and secure communication, key-exchange, public-key encryption, signature, commitment, oblivious transfer, Zero-Knowledge, secret sharing, and general function evaluation. In some cases, we present initial results regarding the realizability of the definitions. In other cases realizing the definitions is left open.

1.1 The proposed framework

We briefly sketch the proposed framework and highlight some of its properties. Let us first briefly sketch the definitional approach of [C00], which is the starting point of this work. (The work of [C00] is, in turn, based to a large extent on [B91, MR91, GL90]). This work is geared towards capturing the task of secure function evaluation in a synchronous, ideally authenticated network. The idea is to first formulate a model representing the process of protocol execution in real-life. This is called the **real-life model**. Next, in order to capture the security requirements of a given task, formulate an **ideal process** for carrying out the task. Then say that a protocol **securely realizes** the task at hand if running the protocol in the real-life model amounts to "emulating" the ideal process for that task.

The real-life model of computation in [C00] consists of a set of interactive Turing machines (ITMs) representing the parties running the protocol, plus an ITM representing the adversary. The parties and adversary interact on a given set of inputs and each party generates local output. The concatenation of the local outputs of all parties and adversary is called the **global output.** The ideal process for evaluating some function f is defined similarly, with the important exception that the parties hand their inputs to an incorruptible *trusted party*, which evaluates f and hands the corresponding outputs back to the parties. A protocol π **securely evaluates** a function f if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that, for any input vector, the global output of running π with $\mathcal{A}$ in the real-life model is indistinguishable from the global output of the ideal process for f with adversary $\mathcal{S}$. This def-

initional approach is sufficient for capturing "stand-alone" security of protocols. It is also shown to be closed under non-concurrent composition.

The present framework preserves the overall structure of that approach. The difference lies in new formulations of the models of computation and the notion of "emulation". Specifically, we introduce an additional computational entity, called the **environment machine**, to both the real-life model and the the ideal process. The environment machine is an ITM that represents "whatever is external to the current protocol execution". This includes other protocol executions and their adversaries, human users, etc. The environment provides all the inputs to all parties and reads all their outputs. More importantly, the environment interacts with the adversary freely throughout the computation. That is, between any two atomic operations carried out by the adversary (e.g., delivery of a message, or corruption of a party) the adversary and the environment may exchange arbitrary information. The security requirement is now that executing the protocol in the real-life model should "look the same" as the ideal process *from the point of view of the environment.* More precisely, a protocol π **securely realizes** a trusted party for some function f if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that *no feasible environment* can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and π in the real-life model or with $\mathcal{S}$ in the ideal process for f. In a way, the environment serves as an "interactive distinguisher" between the protocol execution and the ideal process. Note that the same ideal-process adversary $\mathcal{S}$ is required to work for all environments. Thus, the interaction between $\mathcal{S}$ and the environment is inherently "black-box" from the point of view of $\mathcal{S}$. This requirement is essential for our proof of the composition theorem.[1]

Another modification to the definition allows capturing not only secure function evaluation but also *reactive* tasks where new input values become known throughout the computation, and may depend on previously generated output values. This is obtained by replacing the "trusted party" in the ideal process for secure function evaluation with a general algorithmic entity called an **ideal functionality.** The ideal functionality, which is modeled as another ITM, repeatedly receives inputs from the parties and provides them with appropriate output values. This way, it is guaranteed that the outputs of the parties in the ideal process have the expected properties with respect to the inputs, even when new inputs are chosen adaptively based on previous outputs.

[1] A very limited variant of the notion of environment appears in [C00]. That variant, aimed at providing non-concurrent composition in the presence of an adaptive adversary, interacts with the parties and the adversary only at few occasions throughout the computation. In particular, the variant there is not known to be sufficient for preserving security under concurrent composition.

Yet another difference from [C00] is that here we model networks where the communication is *open, unauthenticated,* and *asynchronous* (without guaranteed delivery of messages). We also concentrate on the case where the adversary is probabilistic polynomial time (PPT). This modeling seems more suitable for analyzing protocols in realistic settings.

Universal Composition. We show that the following property holds with respect to a protocol ρ that securely realizes some ideal functionality $\mathcal{F}$. Let π be some arbitrary protocol (we think of π as an "application protocol") that operates in a model where all parties have ideal access to multiple instances of $\mathcal{F}$. That is, in this model (which we call the $\mathcal{F}$**-hybrid** model) the parties, the adversary and the environment interact as in the real-life model, and in addition the parties can privately communicate with as many instances of $\mathcal{F}$ as they wish. It is stressed that the different instances of $\mathcal{F}$ are running at the same time without any global coordination. They are distinguished via special identifiers, generated by the calling protocol.

Now, construct the composed protocol π^ρ from π by replacing each call to a new instance of $\mathcal{F}$ with an invocation of a fresh copy of ρ. Similarly, a message sent to an existing instance of $\mathcal{F}$ is replaced with an input value given to the corresponding invocation of ρ, and any output of an invocation of ρ is treated as a message received from the corresponding instance of $\mathcal{F}$. (Note that a run of protocol π^ρ may have an unbounded number of copies of ρ which are running concurrently on related inputs.)

The universal composition theorem states that running protocol π^ρ in the plain real-life model has essentially the same effect as running protocol π in the $\mathcal{F}$-hybrid model. More precisely, it guarantees that for any real-life adversary $\mathcal{A}$ there exists an adversary $\mathcal{H}$ in the $\mathcal{F}$-hybrid model such that no environment machine can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running π^ρ in the plain real-life model, or with $\mathcal{H}$ and parties running π in the $\mathcal{F}$-hybrid model. In particular, if π securely realizes some ideal functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{G}$ from scratch.

Notice that, while other composition theorems address only the case where a single protocol instance is composed with another protocol, here the hybrid model allows for an unbounded number of instances of the composed protocol to run concurrently. It may appear that this more complex formulation of the composition operation is not necessary, since it can be obtained by iteratively composing all instances of the protocol, one at a time, with an outside protocol. However, in our computational setting the composition theorem can be safely applied only a *constant* number of times, otherwise the complexity of the adversary $\mathcal{H}$ in the hybrid model may become super-polynomial. Consequently, this weaker formulation of the composition theo-

rem only guarantees secure composition of a *constant* number of protocol instances running concurrently.

Interpreting the composition theorem. Traditionally, composition theorems are treated as tools for modular design and analysis of complex protocols. (For instance, this is the main motivation in [MR91, C00, DM00, PW00].) That is, given a complex task, first partition the task to several, simpler sub-tasks. Then, design protocols for securely realizing the sub-tasks, and in addition design a protocol for realizing the given task in a model where ideal evaluation of the sub-tasks is possible. Finally, use the composition theorem to argue that the protocol composed from the already-designed sub-protocols securely realizes the given task. (An example of a context where this interpretation is put to use is the proof of security in [CKOR00].) Note that for this application it suffices to use composition theorems where it is known in advance which protocol instances are running together and how the protocol executions are going to be interleaved.

In contrast, here we use the composition theorem as a tool for gaining confidence in the sufficiency of a definition of security in some protocol environment. Indeed, protocols that satisfy a universally composable definition are guaranteed to maintain their security within any protocol environment — even environments that are not known a-priori, and even environments where the participants in a protocol execution are unaware of other instances of the protocol (or other protocols altogether) that may be running concurrently in the system in an adversarially coordinated manner. This is a strong guarantee.

1.2 General satisfiability of the definitions

Definitions of security in the proposed framework tend to be more stringent than other definitions. Moreover, the existing proofs of security of many known protocols do not work in the present framework. Examples include most known Zero-Knowledge protocols, the protocol generator of [GMW87, G98] and more. This is mainly due to the fact that a common proof-technique, namely black-box simulation with rewinding of the adversary, does not work in the present framework. (Indeed, here the ideal-process adversary has to interact with the environment machine which cannot be "rewound".)

Nonetheless, it can be seen that some known protocols for the general task of secure function evaluation are, in fact, universally composable. For instance, the [BGW88] protocol (say, with the simplification of [GRR98]), together with encrypting each message using non-committing encryption [CFGN96], is universally composable as long as less than a third of the parties are corrupted. Using [RB89], any corrupted minority is tolerable. The asynchronous setting can be handled using the techniques of [BCG93, BKR94].

We use this fact to demonstrate that practically *any* ideal functionality — even reactive ones and even "two-party functionalities" (i.e., functionalities where only two parties have inputs and outputs) — can be securely realized in the proposed framework. Specifically, our solution assumes that the network contains a set of parties (called "servers") such that only a minority of these parties can ever be corrupted. The servers have no local inputs or outputs; they only assist other parties in realizing the given functionality. All parties share their inputs among the servers, who run the appropriate multiparty function evaluation protocol and send the output values to the appropriate parties. Iterated evaluations (which may involve implementing ideal functionalities that maintain internal state between invocations) are handled in standard ways.

1.3 UC definitions of some specific tasks

We formulate and study universally composable definitions of a number of standard cryptographic tasks. In fact, much of the definitional work is already done by the general framework described above. All that is left to do on the definitional side is to formulate ideal functionalities that capture the security requirements of these tasks.

We first address the task of *message authentication:* the corresponding functionality, $\mathcal{F}_{\text{AUTH}}$, is invoked with a request by some party, P_i, to transmit a message m to another party, P_j. Then $\mathcal{F}_{\text{AUTH}}$ ideally sends (P_i, m) to P_j and the adversary, and halts. (Forwarding (P_i, m) to the adversary captures the fact that secrecy is not provided.) This way, the standard computational model where the communication is ideally authenticated is rephrased as the $\mathcal{F}_{\text{AUTH}}$-hybrid model. This notion is the natural universally composable extension of the *authenticators* of [CHH00, BCK98]. In particular, the two authenticators presented in [BCK98] securely realize $\mathcal{F}_{\text{AUTH}}$ given an authenticated initialization phase.

The task of providing secure (i.e., authenticated *and secret*) transmission of individual messages is addressed next. It is seen that standard semantically secure encryption (or alternatively non-committing encryption for adaptive adversaries) are sufficient in order to realize the secure communication functionality in the $\mathcal{F}_{\text{AUTH}}$-hybrid model, if any message is encrypted using a different public/private key pair.

Next we formulate ideal functionalities that capture the tasks of *secure sessions* and *key exchange.* Secure sessions is an extension of secure transmission of individual messages to the case where a sequence of messages between a pair of parties are secured together. The main advantage of this functionality over the previous ones is that it allows for more efficient realizations, via key-exchange combined with symmetric cryptography using the generated keys. The key exchange functionality essentially provides parties with

"ideally chosen keys." In particular, protocols that securely realize $\mathcal{F}_{\text{KE}}$ are guaranteed to satisfy the security notion of [CK01]. Furthermore, most of the Key-Exchange protocols presented in [CK01] securely realize $\mathcal{F}_{\text{KE}}$.

Next the tasks of *public-key encryption* and *digital signatures* are addressed. Securely realizing the signature ideal functionality turns out to be essentially equivalent to existential security against chosen message attacks as in Goldwasser Micali and Rivest [GMRi88]. In the case of public-key encryption (where many messages may be encrypted by different parties using the same key), securely realizing the proposed functionality turns out to be closely related (but incomparable) to security against adaptive chosen ciphertext attacks [DDN00, RS91, BDPR98].

We then proceed to formulate UC definitions of "classic" two-party primitives such as *coin-tossing, commitment, zero-knowledge,* and *oblivious-transfer.* These primitives are treated as two-party protocols in a multi-party setting. As usual, the composition theorem guarantees that security is maintained under concurrent composition, either with other copies of the same protocol or with other protocols, and within any application protocol. In particular, non-malleability with respect to an *arbitrary* set of protocols is guaranteed. Unfortunately, these functionalities cannot be securely realized by two-party protocols in the bare model of computation (or even in the $\mathcal{F}_{\text{AUTH}}$-hybrid model). This result is shown in [CF01] for the cases of commitment and coin-tossing. Using similar techniques, we extend this result to the cases of oblivious transfer and zero-knowledge. Nonetheless, as demonstrated in [CF01], the commitment and zero-knowledge functionalities can be securely realized by two-party protocols in a hybrid model with ideal access to the coin-tossing functionality. (It is interesting to note that hybrid model with ideal access to the coin-tossing functionality turns out to essentially identical to the popular *common random string model* of [BFM89].)

Finally, we formulate ideal functionalities that capture traditional multi-party tasks such as Verifiable Secret Sharing and Secure Function Evaluation in *synchronous* networks. In particular, we obtain the first definition of protocols for secure function evaluation that is closed under concurrent composition in a setting where all the communication is public. (Two-party Secure Function Evaluation is obtained as a special case.)

1.4 Related work

Numerous definitional works on security of protocols have been carried out over the years. The works of [GL90, MR91, B91, C95] are surveyed in [C00]. Here we very briefly review some definitional efforts that are closely related to the present work.

Pfitzmann et. al. [PW94, PSW00, PSW00a, PW00, PW01] were the first to formally model the security requirements of general reactive systems. In a series of works that contain many interesting ideas, they model security of reactive systems in an extended finite-state machine model of computation that is essentially equivalent to the I/O automata model of [L96]. In particular, they introduce the notion of an **honest user** that 'sees' the functionality (i.e., the inputs and outputs) of a given system, and say that one system "simulates" another if the honest user cannot tell the difference between the two systems. However, they stop short of defining security of protocols for realizing a given task. They also state a composition theorem with respect to their framework; their composition theorem is weaker than the one here in that it deals only with the case where a single protocol execution is carried out concurrently with the calling protocol. (In contrast, much of the complexity in protocol composition appears only when the number of composed copies is not a-priori bounded.) These works contain also descriptions of ideal systems for public-key encryption and certified mail.

Canetti [C00] presents a definition of *secure function evaluation* in a variety of computational settings, including the case where the communication is public and security is guaranteed only against a computationally bounded adversary. Some of the definitions there also use an "environment machine". however, there both the purpose of the environment and its pattern of interaction with the other participants are different than here. The definitions of [C00] are shown to be closed under a composition theorem similar to the one here, but only in the *non-concurrent* case where no more than a single protocol execution is running at any point in time.

Dodis and Micali [DM00] build on the definition of Micali and Rogaway [MR91] for information-theoretically secure function evaluation in *synchronous* networks, where ideally private communication channels are assumed. In that setting, they prove that their definition of security is closed under a general concurrent composition operation similar to the one in this work. They also formulate an additional and interesting composition operation (called *synchronous composition*) that provides stronger security guarantees, and show that their definition is closed under that composition operation in cases where the scheduling of the various invocations of the protocols can be controlled. However, their definition applies only to settings where the communication is ideally private. It is not clear how to extend this definitional approach to realistic settings where the adversary can eavesdrop to the communication between honest parties.

The pioneering work of Dolev, Dwork and Naor [DDN00] points out some important security concerns that arise when running cryptographic protocols within a larger system. In particular, they define and construct encryp-

tion schemes secure against chosen ciphertext attacks, non-malleable commitment schemes, and more. That work provides motivation for the present one. In particular, making sure that the concerns pointed out in [DDN00] are answered plays a central role in the present framework.

Connections with the formal-methods approach to analyzing security. A large body of work on analyzing security of protocols using techniques for formal verification of computer programs has been carried out over the years (a very partial list of works includes [DY83, BAN90, M94, KMM94, L96, AG97]). The approach and framework presented here may serve as a "bridge" for connecting that approach with the complexity-based approach pursued in the cryptographic community, with advantages to both approaches. See more details in our Technical Report [C01].

Organization. Section 2 defines the notion of securely realizing an ideal functionality. Section 3 presents the composition theorem and very briefly outlines the proof. Section 4 states the general satisfiability theorem. Throughout, the presentation is kept high-level and informal for brevity and clarity. Details are available in our Technical Report [C01]. UC definitions of the tasks mentioned in Section 1.3 also appear there.

2 The basic framework

As sketched in Section 1.1, protocols that securely carry out a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in the presence of an adversary and in a given computational environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality", which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process for that ideal functionality. In the rest of this subsection we overview the model for protocol execution (called the *real-life model*), the ideal process, and the notion of protocol emulation.

We concentrate mainly on the following standard model, aimed at representing current realistic communication networks (such as the Internet). The network is *asynchronous* without guaranteed delivery of messages. The communication is public and *unauthenticated.* That is, the adversary may delete, modify, and generate messages at wish. Parties may be broken into (i.e., become corrupted) throughout the computation, and once corrupted their behavior is arbitrary (or, *Byzantine*). Finally, all the involved entities are restricted to probabilistic polynomial time (or, "feasible") computation. Most other standard models of computation (e.g., authenticated communication, synchronous message delivery, the common reference string model, or computationally unbounded adversaries) can be captured via appropriate modifications to the basic model. See more details within.

Protocol syntax. Following [GMRa89, G01], a protocol is represented as a system of interactive Turing machines (ITMs), where each ITM represents the program to be run within a different party. Specifically, the input and output tapes model inputs and outputs that are received from and given to other programs running on the same machine, and the communication tapes model messages sent to and received from the network. Adversarial entities are also modeled as ITMs. We concentrate on a model where the adversaries have an arbitrary additional input, or an "advice". From a complexity-theoretic point of view, this essentially implies that adversaries are non-uniform ITMs.

Protocol execution in the real-life model. We sketch the process of executing a given protocol π (run by parties $P_1, ..., P_n$) with some adversary $\mathcal{A}$ and an environment machine $\mathcal{Z}$ with input z. All parties have a **security parameter** $k \in \mathbf{N}$ and are polynomial in k. The execution consists of a sequence of *activations*, where in each activation a single participant (either $\mathcal{Z}$, $\mathcal{A}$, or some P_i) is activated. The environment is activated first. In each activation it may read the contents of the output tapes of all parties, and may write information on the input tape of either one of the parties or of the adversary. Once the activation of the environment is complete (i,e, once the environment enters a special waiting state), the entity whose input tape was written on is activated next.

Once the adversary is activated, it may read its own tapes and in addition the contents of the outgoing communication tapes of all parties. It may either **deliver** a message to some party by writing this message on the party's incoming communication tape[2], or **corrupt** a party. Upon corrupting a party, the adversary gains access to all the tapes of that party and controls all the party's future actions. Finally, the adversary may write arbitrary information on its output tape. If the adversary delivered a message to some uncorrupted party in an activation then this party is activated once the activation of the adversary is complete. Otherwise the environment is activated next.

Once a party is activated (either due to an input given by the environment or due to a message delivered by the adversary), it follows its code and possibly writes local outputs on its output tape and outgoing messages on its outgoing communication tape. Once the activation of the party is complete the environment is activated. The protocol exe-

[2] In the bare model we do not make any restrictions on the delivered messages. In particular, they need not be related to any of the messages generated by the parties.

cution ends when the environment completes an activation without writing on the input tape of any entity. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z,\vec{r})$ denote the output of environment $\mathcal{Z}$ when interacting with adversary $\mathcal{A}$ and parties running protocol π on security parameter k, input z and random input $\vec{r} = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1 \ldots r_n$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$, $r_{\mathcal{A}}$ for $\mathcal{A}$; r_i for party P_i). Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ denote the random variable describing $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z,\vec{r})$ when $\vec{r}$ is uniformly chosen. Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbf{N},z\in\{0,1\}^*}$.

The ideal process. Security of protocols is defined via comparing the protocol execution in the real-life model to an *ideal process* for carrying out the task at hand. A key ingredient in the ideal process is the **ideal functionality** that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM that interacts with the environment and the adversary via a process described below. More specifically, the ideal process involves an ideal functionality $\mathcal{F}$, an **ideal process adversary** $\mathcal{S}$, an environment $\mathcal{Z}$ with input z, and a set of **dummy parties** $\tilde{P}_1, ..., \tilde{P}_n$.

As in the process of protocol execution in the real-life model, the environment is activated first. As there, in each activation it may read the contents of the output tapes of all (dummy) parties, and may write information on the input tape of either one of the (dummy) parties or of the adversary. Once the activation of the environment is complete the entity whose input tape was written on is activated next.

The dummy parties are fixed and simple ITMs: Whenever a dummy party is activated with input x, it forwards x to the ideal functionality $\mathcal{F}$, say by writing x on the incoming communication tape of $\mathcal{F}$. In this case $\mathcal{F}$ is activated next. Whenever a dummy party is activated due to delivery of some message it copies this message to its output. In this case $\mathcal{Z}$ is activated next.

Once $\mathcal{F}$ is activated, it reads the contents of its incoming communication tape, and potentially sends messages to the parties and to the adversary by writing these messages on its outgoing communication tape. Once the activation of $\mathcal{F}$ is complete, the entity that was last activated before $\mathcal{F}$ is activated again.

Once the adversary $\mathcal{S}$ is activated, it may read its own input tape and in addition it can read the *destinations* of the messages on the outgoing communication tape of $\mathcal{F}$. That is, $\mathcal{S}$ can see the identity of the recipient of each message sent by $\mathcal{F}$, but it cannot see the *contents* of this message (unless the recipient of the message is $\mathcal{S}$). $\mathcal{S}$ may either **deliver** a message from $\mathcal{F}$ to some party by having this message copied the party's incoming communication tape or **corrupt** a party.[3] As usual, upon corrupting a party, the adversary gains access to all the tapes of that party and controls all the party's future actions. If the adversary delivered a message to some uncorrupted (dummy) party in an activation then this party is activated once the activation of the adversary is complete. Otherwise the environment is activated next.

As in the real-life model, the protocol execution ends when the environment completes an activation without writing on the input tape of any entity. The output of the protocol execution is the (one bit) output of $\mathcal{Z}$.

Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z,\vec{r})$ denote the output of environment $\mathcal{Z}$ after interacting in the ideal process with adversary $\mathcal{S}$ and ideal functionality $\mathcal{F}$, on security parameter k, input z, and random input $\vec{r} = r_{\mathcal{Z}}, r_{\mathcal{S}}, r_{\mathcal{F}}$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$, $r_{\mathcal{S}}$ for $\mathcal{S}$; $r_{\mathcal{F}}$ for $\mathcal{F}$). Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ denote the random variable describing $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z,\vec{r})$ when $\vec{r}$ is uniformly chosen. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the ensemble $\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbf{N},z\in\{0,1\}^*}$.

Securely realizing an ideal functionality. We say that a protocol ρ **securely realizes** an ideal functionality $\mathcal{F}$ if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running ρ in the real-life process, or it is interacting with $\mathcal{S}$ and $\mathcal{F}$ in the ideal process. This means that, from the point of view of the environment, running protocol ρ is 'just as good' as interacting with an ideal process for $\mathcal{F}$. (In a way, $\mathcal{Z}$ serves as an "interactive distinguisher" between the two processes. Here it is important that $\mathcal{Z}$ can provide the process in question with *adaptively chosen* inputs throughout the computation.) More precisely, say that a distribution ensemble $X = \{X(k,a)\}_{k\in\mathbf{N},a\in\{0,1\}^*}$ is **binary** if for each k, a the distribution $X(k,a)$ ranges over $\{0,1\}$. Then:

Definition 1 *Two binary distribution ensembles X and Y are* **indistinguishable** *(written $X \approx Y$) if for any $c \in \mathbf{N}$ there exists $k_0 \in \mathbf{N}$ such that for all $k > k_0$ and for all a we have $|\Pr[X(k,a)=1] - \Pr[Y(k,a)=1]| < k^{-c}$.*

Definition 2 *Let $n \in \mathbf{N}$. Let $\mathcal{F}$ be an ideal functionality and let π be an n-party protocol. We say that π* **securely realizes** *$\mathcal{F}$ if for any adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ we have*

$$\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \approx \text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}.$$

Sometimes we will want to restrict the definition so that it quantifies only over adversaries from a certain class (say,

[3] It is stressed that in the ideal process the delivery of messages from $\mathcal{F}$ to the parties is ideally authentic and secret. The adversary only learns that a message was delivered.

the class of adversaries that corrupt only a certain number of parties.) This is done in a straightforward way. See [C01].

Remark: On the requirement to generate output. Recall that the ideal process does not require the ideal-process adversary to deliver messages that are sent by the ideal functionality to the dummy parties. Consequently, the definition provides no guarantee that a protocol will ever generate output or "return" to the calling protocol. (In fact, a protocol that "hangs," never sends any messages and never generates output securely realizes any ideal functionality.) Indeed, in our setting where message delivery is not guaranteed it is impossible to require that a protocol "terminates", or generates output. Instead, the definition concentrates on the security requirements *in the case that the protocol generates output.*

In cases where message delivery is guaranteed (such as in synchronous networks) the ideal process can be modified to exclude protocols that do not generate output. See more details in [C01].

3 The composition theorem

The hybrid model. In order to state the composition theorem, and in particular in order to formalize the notion of a real-life protocol with access to an ideal functionality, the **hybrid** model of computation with access to an ideal functionality $\mathcal{F}$ (or, in short, the $\mathcal{F}$-hybrid model) is formulated. This model is identical to the real-life model, with the following additions. On top of sending messages to each other, the parties may send messages to and receive messages from an unbounded number of copies of $\mathcal{F}$. Each copy of $\mathcal{F}$ is identified via a unique **session identifier** (SID); all messages addressed to this copy and all message sent by this copy carry the corresponding SID. (The SIDs are chosen by the protocol run by the parties.)

The communication between the parties and each one of the copies of $\mathcal{F}$ mimics the ideal process. That is, once a party sends a message to some copy of $\mathcal{F}$, that copy is immediately activated and reads that message off the party's tape. Furthermore, although the adversary in the hybrid model is responsible for delivering the messages from the copies of $\mathcal{F}$ to the parties, it does not have access to the contents of these messages. It is stressed that the environment does not have direct access to the copies of $\mathcal{F}$. (Indeed, here the security definition will require that the environment will be unable to tell whether it is interacting with the real-life model or the hybrid model.)

Replacing a call to $\mathcal{F}$ with a protocol invocation. Let π be a protocol in the $\mathcal{F}$-hybrid model, and let ρ be a protocol that securely realizes $\mathcal{F}$ (with respect to some class of adversaries). The composed protocol π^ρ is constructed by modifying the code of each ITM in π so that the first message sent to each copy of $\mathcal{F}$ is replaced with an invocation of a new copy of π with fresh random input, and with the contents of that message as input. Each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of ρ, with the contents of that message given to ρ as new input. Each output value generated by a copy of ρ is treated as a message received from the corresponding copy of $\mathcal{F}$.

If protocol ρ is a protocol in the real-life model then so is π^ρ. If ρ is a protocol in some $\mathcal{G}$-hybrid model (i.e., ρ uses ideal evaluation calls to some functionality $\mathcal{G}$) then so is π^ρ.

Theorem statement. In its general form, the composition theorem basically says that if ρ securely realizes $\mathcal{F}$ then an execution of the composed protocol π^ρ "emulates" an execution of protocol π in the $\mathcal{F}$-hybrid model. That is, for any real-life adversary $\mathcal{A}$ there exists an adversary $\mathcal{H}$ in the $\mathcal{F}$-hybrid model such that no environment machine $\mathcal{Z}$ can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and π^ρ in the real-life model or it is interacting with $\mathcal{H}$ and π in the $\mathcal{F}$-hybrid model.

A corollary of the general theorem states that if π securely realizes some functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model, and ρ securely realizes $\mathcal{F}$ in the real-life model, then π^ρ securely realizes $\mathcal{G}$ in the real-life model. (Here one has to define what it means to securely realize functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model. This is done in the natural way.)

Theorem 3 (Universal composition) *Let $\mathcal{F}$ be an ideal functionality. Let π be an n-party protocol in the $\mathcal{F}$-hybrid model, and let ρ be an n-party protocol that securely realizes $\mathcal{F}$. Then for any real-life adversary $\mathcal{A}$ there exists a hybrid-model adversary $\mathcal{H}$ such that for any environment machine $\mathcal{Z}$ we have $\text{REAL}_{\pi^\rho,\mathcal{A},\mathcal{Z}} \approx \text{HYB}^{\mathcal{F}}_{\pi,\mathcal{H},\mathcal{Z}}$.*

In particular, if π securely realizes some ideal functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{G}$ from scratch.

Discussion: On the hybrid model. It may seem sufficient at first to formulate a simplified version of the hybrid model and the composition theorem, where only a single copy of the ideal functionality $\mathcal{F}$ is available (instead of multiple copies, as defined above). One could then handle protocols that use multiple copies of $\mathcal{F}$ by repeated applications of the composition theorem until all copies of $\mathcal{F}$ are replaced by calls to a protocol ρ that securely realizes $\mathcal{F}$. This would work even in the concurrent case, since the simplified hybrid model would allow the evaluation of $\mathcal{F}$ to run concurrently with the calling protocol, π.

However, in our setting the composition theorem can be safely repeated only a *constant* number of times. This is so, since the complexity of the hybrid model adversary $\mathcal{H}$ may depend on (in fact, it can be any polynomial in) the complexity of $\mathcal{A}$. Thus, if the theorem is applied a non-constant

number of times, the resulting $\mathcal{H}$ may not be polynomial in the security parameter. Consequently, the above simplified hybrid model would result in a composition theorem that applies only to the limited case where at most a constant number of concurrent executions of a protocol take place. In contrast, the composition theorem here handles polynomially many concurrent executions, and even cases where the number of concurrent executions is determined by the adversary in an adaptive manner.

On the proof of the composition theorem. We briefly outline the main ideas in the proof of the composition theorem. Let $\mathcal{F}$ be an ideal functionality, let π be a protocol in the $\mathcal{F}$-hybrid model, let ρ be a protocol that securely realizes $\mathcal{F}$ and let π^ρ be the composed protocol. Let $\mathcal{A}$ be a real-life adversary, geared towards interacting with parties running π^ρ. We wish to construct an adversary $\mathcal{H}$ in the $\mathcal{F}$-hybrid model such that no $\mathcal{Z}$ will be able to tell whether it is interacting with π^ρ and $\mathcal{A}$ in the real-life model or with π and $\mathcal{H}$ in the $\mathcal{F}$-hybrid model. For this purpose, we are given an ideal-process adversary $\mathcal{S}_\rho$ that works for a single execution of protocol ρ as stand-alone. Essentially, $\mathcal{H}$ will run a simulated copy of $\mathcal{A}$ and will follow the instructions of $\mathcal{A}$. The interaction of $\mathcal{A}$ with the various copies of ρ will be simulated using multiple copies of $\mathcal{S}_\rho$. For this purpose, $\mathcal{H}$ will play the environment for the copies of $\mathcal{S}_\rho$. The ability of $\mathcal{H}$ to obtain timely information from the multiple copies of $\mathcal{S}_\rho$, by playing the environment for them, is at the crux of the proof.

The validity of the simulation is demonstrated via reduction to the validity of $\mathcal{S}_\rho$. Dealing with many copies of $\mathcal{S}_\rho$ running concurrently is done using a hybrids argument in which we define many hybrid executions where in each hybrid execution a different number of copies of the ideal functionality $\mathcal{F}$ are replaced with copies of ρ. This hybrid argument is made possible by the fact that $\mathcal{S}_\rho$ must be defined independently of the environment (this is guaranteed by the order of quantifiers), thus it remains unchanged under the various "hybrid environments".

4 Realizing general ideal functionalities

This section argues that realizing ideal functionalities in the present formalization is, in general, feasible. That is, we demonstrate that a large class of ideal functionalities can be securely realized in some standard settings, using known techniques, as long as the protocol involves a set of parties such that only a certain fraction of these parties are corrupted.

As remarked in Section 2, in models where message delivery is not guaranteed the protocol that generates no messages and no outputs securely realizes any ideal functionality. Consequently, a general feasibility theorem is of no interest in such cases. In contrast, a general feasibility theorem is indeed of interest in settings where message delivery is guaranteed, and protocols are required to generate output. Two important such settings are the cases of *synchronous* networks and *asynchronous networks with guaranteed delivery* The rest of this section concentrates on synchronous networks (see precise definition in [C01]).

We restrict the discussion to *authenticated networks*, i.e. to networks with ideally authenticated communication. (See more discussion and formal definition in [C01].) In this setting, assume that there exists a set A of parties (called "servers") such that less than $|A|/3$ of the servers may be corrupted by the adversary throughout the computation. In this case we show:

Theorem 4 *Consider a synchronous network with a set of servers as described above, and assume that trapdoor permutations exist. Then, for any ideal functionality $\mathcal{F}$ there exists a protocol that securely realizes $\mathcal{F}$ in this network.*

A crucial ingredient in the proposed protocol is to have the protocol include "assistant programs" to be run by the servers. These programs will not have local inputs or outputs; instead, they will be invoked by incoming messages from other parties, and will generate messages to other parties (and other servers). Specifically, we use the construction of [BGW88], with the simplification of [GRR98], and in addition encrypt each message using non-committing encryption [CFGN96].

A sketch of proof is given in [C01]. Let us remark that variants of the above theorem hold in most other standard models of computation. In the case of ideally secure communication, encrypting messages (and the assumption that trapdoor permutations exist) is not necessary. In the case of synchronous networks with a broadcast channel, $t < n/2$ is sufficient using the techniques of [RB89, CDDHR99]. In the case of non-blocking asynchronous networks a similar result holds using the techniques of [BCG93, BKR94].

Acknowledgments. Much of the motivation for undertaking this project, and many of the ideas that appear here, come from studying secure key-exchange protocols together with Hugo Krawczyk. I thank him for this long, fruitful, and enjoyable collaboration. I am also grateful to Oded Goldreich who, as usual, gave me both essential moral support and invaluable technical and presentational advice.

Many thanks also to the numerous people with whom I have interacted over the years on definitions of security and secure composition. A very partial list includes Martin Abadi, Mihir Bellare, Ivan Damgaard, Mark Fischlin, Shafi Goldwasser, Rosario Gennaro, Shai Halevi, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, Phil MacKenzie, Tal Malkin, Cathy Meadows, Silvio Micali, Daniele Micciancio, Moni Naor, Rafi Ostrovsky, Tal Rabin, Charlie Rackoff, Phil Rogaway, Victor Shoup, Paul Syverson and Michael Waidner. In particular, Daniele made several observations that have considerably simplified and improved the presentation.

References

[AG97] M. Abadi and A. D. Gordon, A calculus for cryptographic protocols: The spi calculus. In *4th ACM CCS*, 1997, pp.36-47. Fuller version available at

http://www.research.digital.com/SRC/ abadi.

[B91] D. Beaver, "Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority", J. Cryptology, Springer-Verlag, (1991) 4: 75-122.

[B96] D. Beaver, "Adaptive Zero-Knowledge and Computational Equivocation", *28th STOC*, 1996.

[BH92] D. Beaver and S. Haber, "Cryptographic protocols provably secure against dynamic adversaries," *Eurocrypt '92*, 1992, pages 307–323.

[BCK98] M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols", *30th STOC*, 1998.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes", *CRYPTO '98*, 1998, pp. 26-40.

[BR93] M. Bellare and P. Rogaway, "Entity authentication and key distribution", *CRYPTO'93*, 1993.

[BCG93] M. Ben-Or, R. Canetti and O. Goldreich, "Asynchronous Secure Computations", *25th STOC*, 1993, pp. 52-61.

[BGW88] M. Ben-Or, S. Goldwasser and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", *20th STOC*, 1988, pp. 1-10.

[BKR94] M. Ben-Or, B. Kelmer and T. Rabin, Asynchronous Secure Computations with Optimal Resilience, *13th PODC*, 1994, pp. 183-192.

[BFM89] M. Blum, P. Feldman and S. Micali, "Non-interactive Zero-Knowledge and its applications," *20th STOC*, 1988.

[BCC88] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, Vol. 37, No. 2, pages 156–189, 1988.

[BAN90] M. Burrows, M. Abadi and R. Needham, "A logic for authentication," DEC Systems Research Center Technical Report 39, February 1990. Earlier versions in *The Second Conference on Theoretical Aspects of Reasoning about Knowledge*, 1988, and *The Twelfth ACM Symposium on Operating Systems Principles*, 1989.

[C95] R. Canetti, "Studies in Secure Multi-party Computation and Applications",*Ph.D. Thesis*, Weizmann Institute, Israel, 1995.

[C00] R. Canetti, "Security and composition of multi-party cryptographic protocols", *Journal of Cryptology*, Vol. 13, No. 1, winter 2000.

[C01] R. Canetti, "Universally Composable Security: A new paradigm for cryptographic protocols," ECCC TR 01-16. Also available at http://eprint.iacr.org/2000/067. (A previous version of this work was entitled "A unified framework for analyzing cryptographic protocols.")

[CF01] R. Canetti and M. Fischlin, "Universally Composable Commitments", *Crypto 01*, 2001.

[CK01] R. Canetti and H. Krawczyk, "Analysis of key exchange protocols and their use for building secure channels", *Eurocrypt '01*, 2001.

[CFGN96] R. Canetti, U. Feige, O. Goldreich and M. Naor, "Adaptively Secure Computation", *28th STOC*, 1996. Fuller version in MIT-LCS-TR #682.

[CHH00] R. Canetti, S. Halevi and A. Herzberg, "How to Maintain Authenticated Communication", *Journal of Cryptology*, Vol. 13, No. 1, winter 2000. Preliminary version at *16th PODC*, 1997, pp. 15-25.

[CKOR00] R. Canetti, E. Kushilevitz, R. Ostrovsky and A. Rosen, "Randomness vs. Fault-Tolerance", *Journal of Cryptology*, Vol. 13, No. 1, winter 2000. Preliminary version at *16th PODC*, 1997, pp. 35-45.

[CDDHR99] R. Cramer, I. Damgaard, S. Dziembowski, M. Hirt and T. Rabin ,"Efficient multiparty computations secure against an adaptive adversary", *Eurocrypt*, 1999, pp. 311-326.

[DIO98] G. Di Crescenzo, Y. Ishai and R. Ostrovsky, Non-interactive and non-malleable commitment, *30th STOC*, 1998, pp. 141-150.

[DM00] Y. Dodis and S. Micali, "Secure Computation", *CRYPTO '00*, 2000.

[DDN00] D. Dolev, C. Dwork and M. Naor, Non-malleable cryptography, *SIAM. J. Computing*, Vol. 30, No. 2, 2000, pp. 391-437. Preliminary version in *23rd STOC*, 1991.

[DY83] D. Dolev and A. Yao, On the security of public-key protocols, *IEEE Transactions on Information Theory*, 2(29), 1983.

[DNRS99] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer, "Magic functions," *40th FOCS*, pp. 523–534, 1999.

[DNS98] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. *30th STOC*, pages 409–418, 1998.

[EGL85] S. Even, O. Goldreich and A. Lempel, "A randomized protocol for signing contracts", *CACM*, vol. 28, No. 6, 1985, pp. 637-647.

[F91] U. Feige. Ph.D. thesis, Weizmann Institute of Science, 1991.

[FF00] M. Fischlin and R. Fischlin, "Efficient non-malleable commitment schemes", *CRYPTO '00, LNCS 1880*, 2000, pp. 413-428.

[GM00] J. Garay and P. MacKenzie, "Concurrent Oblivious Transfer", *41st FOCS*, 2000.

[GM95] R. Gennaro and S. Micali, Verifiable Secret Sharing as Secure Computation, *Eurocrypt 95, LNCS 921*, 1995, pp. 168-182.

[GRR98] R. Gennaro, M. Rabin and T Rabin, Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography, *17th PODC*, 1998, pp. 101-112.

[G01] O. Goldreich, *"Foundations of Cryptography,"* Springer-Verlag, 2001.

[G98] O. Goldreich. *"Secure Multi-Party Computation"*, 1998. (Available at http://philby.ucsd.edu)

[GK88] O. Goldreich and H. Krawczyk, "On the Composition of Zero-Knowledge Proof Systems," *SIAM. J. Computing*, Vol. 25, No. 1, 1996.

[GMW87] O. Goldreich, S. Micali and A. Wigderson, "How to Play any Mental Game", *19th STOC*, 1987, pp. 218-229.

[GO94] O. Goldreich and Y. Oren, "Definitions and properties of Zero-Knowledge proof systems", *Journal of Cryptology*, Vol. 7, No. 1, Springer-Verlag, 1994, pp. 1–32. Preliminary version by Y. Oren in *28th FOCS*, 1987.

[GL90] S. Goldwasser, and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", *Crypto '90*, 1990.

[GM84] S. Goldwasser and S. Micali, "Probabilistic encryption", *JCSS*, Vol. 28, No 2, April 1984, pp. 270-299.

[GMRa89] S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", *SIAM Journal on Comput.*, Vol. 18, No. 1, 1989, pp. 186-208.

[GMRi88] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, April 1988, pages 281–308.

[HM00] M. Hirt and U. Maurer, "Complete characterization of adversaries tolerable in secure multi-party computation", *Journal of Cryptology*, Vol 13, No. 1, 2000, pp. 31-60. Preliminary version in *16th PODC*, 1997, pp. 25–34.

[KMM94] R. Kemmerer, C. Meadows and J. Millen, Three systems for cryptographic protocol analysis, *J. Cryptology*, 7(2):79-130, 1994.

[LMMS98] P. Lincoln, J. Mitchell, M. Mitchell, A. Schedrov, "A Probabilistic Polytime Framework for Protocol Analysis", *5th ACM CCS*, 1998, pp. 112-121.

[LMMS99] P. Lincoln, J. Mitchell, M. Mitchell, A. Schedrov, "Probabilistic Polynomial-time equivalence and security analysis", *Formal Methods Workshop*, 1999. Available at ftp://theory.stanford.edu/pub/jcm/papers/fm-99.ps.

[L96] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR, *2nd International Workshop on Tools and Algorithms for the construction and analysis of systems*, Springer-Verlag, 1996.

[L96] N. Lynch, *Distributed Algorithms*, Morgan Kaufman, San Francisco, 1996.

[M94] C. Meadows, Formal verification of cryptographic protocols: A survey, *Asiacrypt '94*, LNCS 917, 1995, pp. 133-150.

[MR91] S. Micali and P. Rogaway, "Secure Computation", unpublished manuscript, 1992. Preliminary version in *CRYPTO '91, LNCS 576*, Springer-Verlag, 1991.

[NY90] M. Naor and M. Yung, "Public key cryptosystems provably secure against chosen ciphertext attacks", *22nd STOC*, 427-437, 1990.

[PW94] B. Pfitzmann and M. Waidner, "A general framework for formal notions of secure systems", Hildesheimer Informatik-Berichte 11/94, Universitat Hildesheim, 1994. Available at http://www.semper.org/sirene/lit.

[PSW00] B. Pfitzmann, M. Schunter and M. Waidner, "Secure Reactive Systems", IBM Research Report RZ 3206 (#93252), IBM Research, Zurich, May 2000.

[PSW00a] B. Pfitzmann, M. Schunter and M. Waidner, "Provably Secure Certified Mail", IBM Research Report RZ 3207 (#93253), IBM Research, Zurich, August 2000.

[PW00] B. Pfitzmann and M. Waidner, "Composition and integrity preservation of secure reactive systems", *7th ACM CCS*, 2000, pp. 245-254.

[PW01] B. Pfitzmann and M. Waidner, "A model for asynchronous reactive systems and its application to secure message transmission", IEEE Symposium on Security and Privacy, May 2001. Preliminary version in http://eprint.iacr.org/2000/066, 2000.

[R81] M. Rabin, "How to exchange secrets by oblivious transfer", Tech. Memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.

[RB89] T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multi-party Protocols with Honest Majority", *21st STOC*, 1989, pp. 73-85.

[RS91] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", *Crypto '91*, 1991.

[Sh99] V. Shoup, "On Formal Models for Secure Key Exchange", manuscript, 1999. Available at: http://www.shoup.org.