Electronic Notes in Theoretical Computer Science 32 (2000)
URL: http://www.elsevier.nl/locate/entcs/volume32.html 19 pages

Cryptographic Security of Reactive Systems (Extended Abstract)

Birgit Pfitzmann, Matthias Schunter [1]

Fachrichtung Informatik
Saarland University
Saarbrücken, Germany

Michael Waidner

IBM Zurich Research Laboratory
Rueschlikon, Switzerland

Abstract

We describe some general relations between cryptographic and abstracted security definitions, and we present a novel model of security for reactive systems, generalizing previous definitions relying on the simulatability paradigm.

The larger context is the goal to provide cryptographic semantics for "abstract" specifications, so that the "reality" of the former can be combined with the brevity or, if a formal language is used, the precision and tool-support, of the latter.

The novel aspects of our specific definition are a separate treatment of honest users, a precise synchronous switching model, and easy inclusion of various trust models. We also believe to have the first general strategy to deal abstractly with accepted vulnerabilities (such as leakage of traffic patterns), and the first worked-out serious-size examples within a general model. Most importantly, our model has the first general composition theorem, and a link to requirements formulated in logics.

1 Introduction

Security proofs for systems involving cryptography are getting increasing attention in theory and practice, and they are used for increasingly large systems. While for some time most of the effort concentrated on primitives like encryption and signature schemes themselves, or authentication and key exchange,

[1] This work was partially supported by the MAFTIA project.

currently work is under way on entire secure channels, fair exchange protocols, payment systems, and anonymity systems. In the future, one might want to prove even larger systems like entire electronic-commerce architectures, or distributed operating systems using cryptography.

Two communities are working on such proofs, and the techniques are almost completely disjoint. One of our goals with this paper is to show how to link them, and that one actually needs to link them to get the best possible overall results.

We will therefore start with a rather global introduction. More closely related literature is discussed in Section 2.

1.1 Abstracted Methods

In the formal-methods community, one tries to use established specification techniques to specify requirements and actual protocols unambiguously and with a clear semantics. Moreover, most work aims at proofs that are at least automatically verifiable and if possible automatically constructed.

One always needs some security-specific work, e.g., to model an adversary controlling the network and to formalize specific security requirements in the given language. The earliest work on tool support was rather specific, e.g., the Interrogator [28] and the NRL protocol verifier [25], while nowadays most work is based on standard languages like CSP, e.g., [36,24], or the π-calculus [2] and general-purpose verification tools.

Where one aims at tool-supported proofs, cryptographic primitives are almost always abstracted from in a way introduced in [12]:[2] Each cryptographic operation is treated as a basic operation in an abstract data type or a term algebra. For instance, there is a pair of operators E_X and D_X for asymmetric encryption and decryption with a key pair of a participant X. The result of two encryptions of a message m from a basic message space M is not represented as another message from M, but as the term $E_X(E_X(m))$. One then defines cancellation rules, in particular $D_X(E_X(t)) = t$ for all terms t. A basic assumption for proofs in such a model is that equations that cannot be derived from the explicitly given ones do not hold. Hence, in terms of abstract data types, one works in an initial model of the given formulas. (This model not only underlies papers like the ones cited above, but also the semantics of BAN logic [3,41].)

In most other applications of formal methods, one does not restrict oneself to initial models, but in security, one needs them because one typically wants to show *in*equalities, e.g., that the set of messages an adversary can learn does *not* contain a certain secret message.

A problem with all these models is the missing link between the chosen abstractions and the real cryptographic primitives as defined and sometimes proven in cryptography. Hence all the techniques, even actual proofs in the

[2] Other approaches closer to cryptography are treated in Sections 1.6 and 2.1.

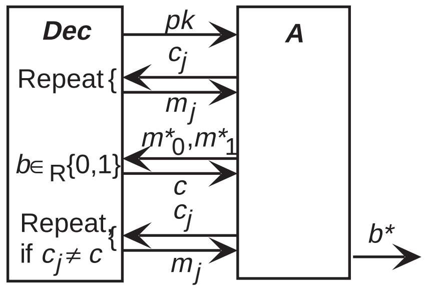

Fig. 1. Cryptographically secure asymmetric encryption

abstraction (no restriction to a finite number of protocol executions etc.), are only a way of rapidly discovering *some* errors with respect to cryptographic reality. We will come back to this with small (and "made-up") examples after presenting a cryptographic definition.

1.2 Cryptographic Definition of Asymmetric Encryption

For comparison with the abstraction above, let us present the cryptographic definition of secure asymmetric encryption. It comprises adaptive chosen-message attacks and secrecy of any partial information about the encrypted message. This is the strongest definition considered in cryptography. Several different definitions, both concerning partial information and active attacks, have all been proven equivalent [26,6]. Hence cryptographers are quite satisfied to have captured the concept adequately. Efficient systems provably secure in this strong sense under reasonable assumptions are known [11].

One defines an encryption system as a triple of polynomial-time algorithms (gen, E, D), where gen and E are probabilistic. On input a security parameter k, gen outputs a key pair (sk, pk). E and D are encryption and decryption, and an equation denoting correct encryption is required as above, i.e., for all key pairs (sk, pk) from the range of gen, and all messages from a certain message space, $D(sk, (E(pk, m)) = m$.

For the security, one considers the following interaction between an arbitrary probabilistic polynomial-time interactive adversary A and an attacked participant Dec ("decryptor"), see Figure 1.[3] Initially, the attacked participant generates a key pair and sends pk to A. The adversary may then ask Dec to decrypt chosen ciphertexts c_j. At some point, A instead chooses two messages m_0^*, m_1^* and Dec encrypts a randomly chosen one of them. The choice is indicated by a bit b, i.e., $c = E(pk, m_b^*)$.[4] Then A may continue the chosen-ciphertext attack, except that Dec now refuses to decrypt the specific

[3] The underlying computational model for both is interactive probabilistic Turing machines [18].

[4] The choice of two "favorite" messages by the adversary models the secrecy of any partial information. For instance, if the adversary can evaluate a particular partial information function on encrypted messages, e.g., their Jacobi symbol as with pure RSA, he would choose m_0^*, m_1^* with different values under this function and thus find out b.

ciphertext c. Finally, A outputs a bit b^* as its guess at b.

The definition is that the probabilities $P_A(k)$ that $b^* = b$ for such an attacker A and the security parameter k should only be negligibly larger than 1/2. This is called the class "1/2 + 1/poly(k)". It means that for any polynomial Q there exists a parameter $k_0 \in \mathbb{N}$ such that for all $k > k_0$,

$$P_A(k) \leq \frac{1}{2} + \frac{1}{Q(k)}.$$

The probability is taken over all probabilistic choices of *Dec* and A, in particular the key generation, the probabilism in the encryption function and the adversary's probabilistic choices.

1.3 Comparison

Obviously, the definition of asymmetric encryption in cryptography and its abstraction are not very similar. We do not even mainly mean obvious differences like the explicit restriction to polynomial-time adversaries and the error probability $1/Q(k)$—these are facts from which you might reasonably want to abstract. The main point is that this definition makes no attempt to cover *all* inequalities that are assumed in the abstraction; it concentrates fully on knowledge about the cleartext given the ciphertext.

For example, a protocol might use a hash function H and attach $H(m)$ to a message m for redundancy in an integrity check. Then tests of the form $tail(m) = H(head(m))$ occur, where *tail* extracts the end of the message of appropriate length for hash values and *head* takes the rest. Now consider the equation

$$tail(E(m)) = H(head(E(m))).$$

In the abstraction, one would certainly not add it to the model, because it does not hold for most encryption systems. Nevertheless, one can easily construct *some* cryptographically secure encryption systems where this equation always holds: Simply take any secure encryption system and augment all ciphertexts c by $H(c)$. It is easy to see that this has no influence on the security in the cryptographic sense: $H(c)$ gives no new information about the encrypted message because it is a function of the ciphertext that the adversary knows anyway. This can be turned into a rigorous cryptographic proof without problems.

This example already shows that the abstract model is not necessarily faithful with respect to the cryptographic semantics of the primitives. We are not aware of a real protocol that fails for such reasons (but we have not searched), but one could make up artificial ones that do. For instance, let us build a five-message fair exchange protocol of a payment promise against something.[5] First, participant X signs a message m_1 that contains a special

[5] Fair exchange means that either both parties should obtain the other's item or none. Here we sketch an optimistic protocol, where a third party can be called upon in incorrect protocol runs to restore fairness, typically if the last messages is not sent.

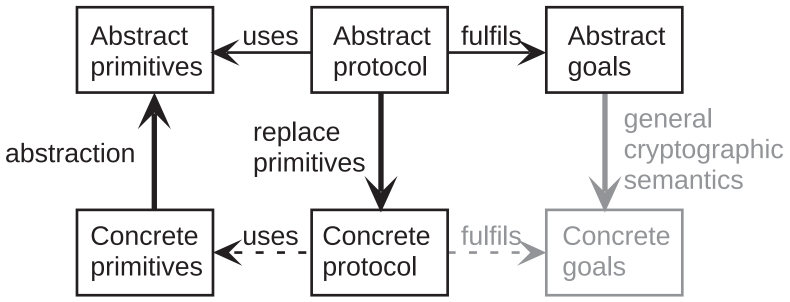

Fig. 2. Faithful abstraction. Bold arrows should be defined once and for all, normal ones have to be provided once per protocol. Dashed arrows should then follow automatically. The gray part does not really exist yet.

signature test key pk' for this protocol run and says that the promise will hold if any message of the form $(m, H(m))$ is signed with respect to pk'. X will actually sign such a message m_5 in the last protocol round. In between, however, he sends some encrypted value $E(m_3)$, also signed with sk'. Now, an adversary can use $sign(sk', E(m_3))$ as $sign(sk', m_5)$ because the only test made on m_5 is $tail(m_5) = H(head(m_5))$. Hence he can hold the honest participant to the promise from m_1 without having sent the message m_4 that he was supposed to send before m_5 for fairness. This problem will not be found in the abstract model.

Of course, this is a stupid protocol, and we have not even worked it out fully. The goal is not to show that something is actually "wrong", but to point out that we have no guarantee that it is "right" in a certain sense that would be nice to have. This is shown in Figure 2. The top layer shows the abstraction; there are protocols built from abstracted primitives and proven secure with respect to certain abstract goals. The abstract primitives represent concrete primitives, and there should be an unambiguous mapping for deriving a concrete protocol from the abstract one by replacing the primitives. (This is more or less clear in the models mentioned above, except for some logics of authentication.) We cannot expect the concrete protocol to fulfil the abstract goals, as long as those abstract from error probabilities and computational restrictions on the adversary (which we find a good thing). However, we would like it to fulfil a concrete version of the goals which is automatically derived from the abstract one by adding similar error probabilities and computational restrictions. This part does not exist yet—neither the cryptographic semantics of general abstract goals, nor the proof that the abstract "fulfils"-relation implies the cryptographic one.

In the following subsections, we discuss previous approaches that can be seen in relation to the gray part of Figure 2. We then present (as extended abstract) a model that fills some of the existing gaps.

1.4 Other Cryptographic Definitions Relating to Abstractions

For certain primitives the approach from the previous section has already been taken to some extent: Mere integrity primitives like signature schemes and symmetric authentication codes can be specified quite well in temporal logic, see [29]; such specifications where also made in other contexts, in particular first in [39]. The special point in [29,31] is that a concrete cryptographic semantics for *general* linear temporal-logic formulas defining safety properties was sketched (in a meta-model, not a concrete underlying machine model). It was shown that applying this model to certain primitives actually gives the cryptographic definition, and that logic implications can be made and carry over to the cryptographic layer.

Another approach that can be interpreted as basing cryptographic definitions on abstract specifications is the simulatability approach at defining secure function evaluation. There, the abstract specification is simply a function f on many inputs $x_1, \ldots, x_n$. The goal of the cryptographic protocol is that each participant provides one x_i as secret input and they all learn the result $y = f(x_1, \ldots, x_n)$, but no further information about the inputs of the other parties. It was quite an effort to actually define what this means in a cryptographic sense, starting with [42], over approaches that tried to define the integrity and privacy of such protocols separately, back to formalizations that more closely resembled the original idea that the protocol should be "just as good as" a trusted host that simply computed f for the participants [17,4,27,8]. This is called the simulatability approach.

Our definition follows this simulatability approach, extending it to more general abstract specifications. More closely related papers that also extend this approach beyond function evaluation are discussed in Section 2.

1.5 What does Cryptography Gain from Abstractions?

So far we have argued why formal methods could benefit from cryptographic semantics. Cryptographers ask the reverse question: Aren't our mathematically rigorous proofs better than any abstraction? I.e., why have an upper layer in Figure 2 at all? The answer is that indeed one does not gain expressiveness and the overall results cannot get more rigorous than done by hand, but the specifications can get nicer, the proofs shorter, and tool support would be easier for those parts of proofs that are accessible to it. Indeed cryptographic definitions are very long (that in Section 1.2 is one of the simplest), and many have been found faulty or at least have been strengthened later. Similarly, most proofs are currently actually sketches, and again some have contained gaps. This will get much worse with larger systems.

1.6 Formal Methods without Abstraction

Another research direction is to try and express *actual* cryptographic definitions in formal languages. This is almost orthogonal to our own goal, which is to provide a class of abstractions with a general cryptographic semantics. I.e., you can have both abstracted and non-abstracted definitions in either normal mathematical or formal languages.

For instance, [20] (modeling security in CSP) and [21] contain such aspects, or, from another direction [40]. More recent examples are [37,22]. Note, however, that most approaches in this field either do not capture the entire real cryptographic definition (e.g., only define that many traces should be compatible with the adversary's view, but nothing about the probabilities), or that only the system model is formal, while the probabilistic part is still a special new semantics in normal mathematical language. Moreover, there is so far not much tool support in this area, in contrast to Section 1.1.

Yet another approach is to add details like homomorphic properties of low-level cryptographic primitives (pure RSA is not at all a secure encryption system in the cryptographic sense) to the abstract data types from Section 1.1, as initiated in [13]. However, there is then still an initial-algebra semantics for the properties one has not added, which so far has no cryptographic justification.

2 Reactive Simulatability Definition

As mentioned above, a main hindrance for making abstract security proofs that are faithful with respect to real cryptographic implementations is that there is no general notion in cryptography yet for the security of an arbitrary protocol if secrecy is involved. We will provide this for one class of specifications by extending the simulatability definition from function evaluation to general reactive systems.

2.1 Related Work and New Aspects in Our Approach

The first sketch of a simulatability definition for general reactive systems was given in [16] (Section 6). In [15] (Section 4.1), and also [14], where such a model was first applied to a concrete protocol, the authors interpret this as including the exact simulation of an internal state. In contrast, we concentrate on comparison of the interface behaviour, i.e., on possible external observations of the system. This was first sketched in [34] (presented in [30,35]). A similar sketch can be found in [9].

Specific definitions in a similar setting were worked out in [5,38,10], i.e., they also rely on simulatability, but are "hand-made" for one specific specification. Furthermore, due to the specific protocol class considered, not all problems of the general case occur.

Detailed general definitions first exist in [19,22]. The former are not for completely general specifications (straight-line programs)—the main goal was to present a general technique to construct secure protocols for any permitted specification—and only for information-theoretic security (in that case, the simpler user model in that paper is probably equivalent to ours below). Also our timing will be different. In [22,23], a simulatability definition was given with π-calculus processes instead of the usual interactive probabilistic Turing machines. Unfortunately none of the prior literature (not even the well-known papers on simulatability for function evaluation from Section 1.4) was cited. While having merits as the first worked-out version with general reactive systems, and the first with a formal language as machine model, the actual definition, indistinguishability of two views, seems standard to us and not within the formal system. Moreover, the specifications lack abstraction. For instance, the specification of secure transmission of a message in [22] consists of the protocol, only all encrypted messages are replaced by random messages encrypted with the same cryptosystem. In contrast, we aim at specifications that are considerably simpler than the actual protocols and protocol-independent.

The main novelties of our model are our separate treatment of machines, honest users, and adversaries (this is already in [34]); a precise synchronous execution model for this case; and easy inclusion of various trust models. Further novelties in our work are:

- We have theorems relating several variants of the definition, trying to sort out which design decisions really change the expressiveness.
- A composition theorem, see Section 4.
- A strategy to deal abstractly with certain non-cryptographic imperfections of most implementations, e.g., the possibility of traffic analysis.
- Serious-sized examples fully worked out within a general model (see Section 3).

Our current definitions are in a synchronous model of time. This can be seen as a restriction compared with asynchronous models, and indeed we first did this to avoid the issue of scheduling between adversary and honest users, which is also not fully worked out in any other model we are aware of. (Of course this should be future work.) However, we believe that a synchronous model is not just a weaker version of an asynchronous one, but any definitions should be made for both. One reason is that there are many synchronous protocols around because one can often gain efficiency in a synchronous model. Secondly, most asynchronous protocols at least need a possibility for user timeouts in reality anyway. Thirdly, if one entirely abstracts from timing (e.g., by assuming that the adversary does it all), one cannot discover certain information leakage resulting from timing. In this sense, a synchronous model that imposes strict timing requirements on correct machines, and includes observations of reaction speeds with respect to this scale, is closer to reality.

We also do not cover dynamic corruptions at the moment.

2.2 Definitions

We now briefly go through our basic definitions. The underlying machine model is standard probabilistic extended finite-state (PEFS) machines with a finite set of in- and output ports (formally a pair of a name and a direction). Where computational statements must be made, we assume that the machines are realized by interactive Turing machines, and the notion "polynomial" is measured in terms of the length of the *initial* inputs, i.e., the initial content of the worktape (all as in [18]).

Definition 2.1 (Structures and Systems) A cryptographic system Sys is modeled as a set of structures. A structure is defined as a triple (M, G, s). Here M is a set of PEFS machines, called correct machines, G is a graph on ports, called connection graph, and s a set of ports, called specified ports. All ports of M should occur in an edge (this is w.l.o.g.). Each edge of G connects one out- and one input port, and each input port occurs only in one edge.[6] Each multicast connection contains at least one port from M. s is a subset of the free ports of G, i.e., those that do not occur in M. These free ports are denoted $free(G, M)$.

This is illustrated together with the following definition in Figure 3. Typically, a cryptographic system is described by an *intended structure*, and the actual structures are derived using a *trust model.* E.g., in a multi-party protocol with honest majority any $k > n/2$ of the intended machines may be present in M (the others are subsumed by the adversary), or in a fair exchange protocol the notary and either of the two exchanging parties. Similarly, the actual channels are derived from the intended ones, depending on whether the intended channels are private, private and authentic, broadcast etc. We have concrete derivation rules for all these standard trust models, but as one can imagine a wide range of trust models we made the above definition generic.

Definition 2.2 (Configuration) A configuration of a cryptographic system Sys is a tuple (M, G, s, H, A, G_{AH}) where $(M, G, s) \in Sys$ is a structure, A and H are machines modeling the adversary and the honest users, and G_{AH} is an additional graph with connections only between A and H. A and H should use all ports from $free(G, M)$, and no ports of A and H should remain free (both w.l.o.g.).

Typically, all ports from s are attached to H and all other ports from $free(G, M)$ to A, i.e., the honest users use precisely the specified ports and the adversary the rest, e.g., wiretaps. We could actually require this for all cryptographic examples we have considered so far, but not for all examples from fault-tolerance.[7] We find this model with specific honest users much

[6] Thus there are internal channels among machines from M and connections between M and an environment. Multicast is modeled by several edges from the same output port.

[7] We could also add a special free output port *guess* to A, which models A's knowledge, and later include it in the comparison in Definition 2.4. But we can show that this is equivalent

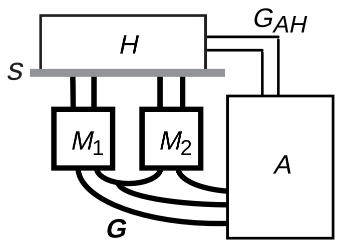

Fig. 3. Configuration of a system. The bold part is the structure.

more intuitive than prior models without: Honest users should not be modeled as part of the machines in M because they are arbitrary, while the machines have prescribed programs. For example, they may have arbitrary strategies which message to input next to the system after certain outputs. They may also be influenced in these choices by the adversary, e.g., in chosen-message attacks on a signature scheme; therefore the graph G_{AH}. Honest users are not a natural part of the adversary either because they are supposed to be protected from the adversary. In particular, they may have secrets, and we want to define that the adversary learns nothing about those except what he learns "legitimately" from the system (this depends on the specification) or what the user tells him directly (via the graph G_{AH}).

We can show that several different versions of the following definition are equivalent (see [32]), but we have not found one that does not need a distinction between specified ports s and other free ports.

For a configuration, we define probability spaces on the runs (or executions, or traces). This would be standard for PEFS machines, except for the fact that adversaries and users cannot be expected to be synchronized with the system rounds. For the adversaries, this is the well-known model of "rushing adversaries"; for users, it is new. As a worst case, one might have to define that adversaries and honest users carry out an arbitrary dialogue within each round, but we can show that this is equivalent to our following simpler model.

Definition 2.3 (Runs and Views) Given a configuration *conf* and an initial input for each machine, a probability space of runs is defined as follows: Each round i has four subrounds, and the switching order is $MH-A-H-A$. This means that in Subround $i.1$, the machines from M and H switch, in Subround $i.2$ the adversary A, etc. The network transports messages according to the graph between any two subrounds; if H or A input two messages per round on a channel to M, M receives their concatenation.

The view of any subset N of the machines can be described by their initial inputs, the random values they used and the messages they received. It is a random variable in the space of the runs and written as $view_{conf,in}(N)$, where in denotes the initial inputs. The family of these random variables indexed by in is called $view_{conf}(N)$.

to our simpler definition.

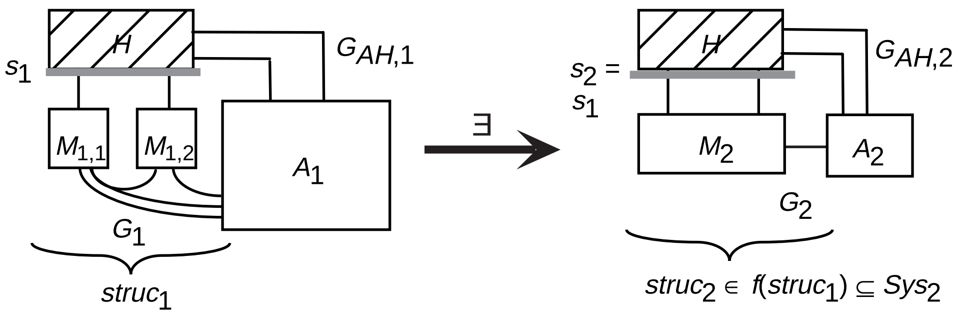

Fig. 4. Simulatability definition with honest users in the simple case $s_1 = s_2$. The hatched part denotes the view of H, which should be indistinguishable.

In the following, we need the cryptographic definition of indistinguishability of families of random variables, see, e.g., [18]. Perfect indistinguishability is simply equality; computational indistinguishability is quite similar to the definition of asymmetric encryption: The adversary is given an element from one of the families and should not be able to guess which one.

For simplicity, we also assume that the initial inputs of all machines are only a security parameter k. The following definition is illustrated in Figure 4.

Definition 2.4 (Simulatability) Let two systems Sys_1 and Sys_2 be given and a function f that maps structures $struc_1$ of Sys_1 to sets of "corresponding" structures of Sys_2.[8] For corresponding structures we require $free(G_1, M_1) \cap free(G_2, M_2) \subseteq s_1 \cap s_2$, i.e., free ports should only have the same name if they are supposed to correspond to each other.

a) We say that Sys_1 is *perfectly at least as secure as* Sys_2 iff for all configurations $conf_1 = (M_1, G_1, s_1, H, A_1, G_{AH,1})$ with $struc_1 = (M_1, G_1, s_1) \in Sys_1$, there exists a configuration $conf_2 = (M_2, G_2, s_2, H, A_2, G_{AH,2})$ with the same user H and $struc_2 = (M_2, G_2, s_2) \in f(struc_1)$ such that

$$view_{conf_1}(H) = view_{conf_2}(H),$$

unless H has port names from $free(G_2, M_2) - s_2$.

b) We say that Sys_1 is *computationally at least as secure as* Sys_2 iff the same as in a) holds for polynomial-time users and adversaries, and with computational indistinguishability of the families of views instead of equality.

Typically, Sys_1 is a "real" system and Sys_2 an "ideal" system used as a specification. Formally, we make no special requirements on ideal systems, but often each of their structures contains only one machine (called "ideal host" in cryptography), while the real system is distributed.

The fact that users H that use ports from $free(G_2, M_2) - s_2$ are excluded is where the distinction of users and adversary makes a real difference. It means

[8] In cryptographic examples, $f(struc_1)$ is typically the set of all structures with the same specified ports s, but not in all fault-tolerance examples.

that the ideal system may have channels to the adversary and thus make certain events visible or manipulatable in an abstract way that are not visible to normal users. This allows us to model systems with certain accepted, a-priori known vulnerabilities, which is important in practice. If we would allow H access there, these exact events would have to be indistinguishable from some events in the real system, which would seriously limit the possible abstraction. This should become clear in the following example.[9]

3 Examples and Accepted Vulnerabilities

We have so far worked out two examples in this model in full detail, a protocol for computationally secure (private and authentic) message transmission and one for optimistic certified mail. The former is similar to examples already considered with a simulatability approach (but not in a general model), the latter is new. Both consider an arbitrary number of participants that can run many instances of the basic protocol, because only then typical protocol failures show up. In both cases we have a protocol-independent specification, and then a proof for one specific protocol.

We can only sketch the first example here to show how our definition with honest users permits easy and truly abstract specifications, in particular in connection with accepted vulnerabilities as one often finds them in real-world applications. (See [33] for the second example.)

The nicest abstract model for secure message transmission would be one machine that models a perfect network, i.e., users can input messages with addresses, and the messages are delivered precisely to these addresses in the same order. However (as also observed, e.g., in [1]), this would not allow implementations by normal use of encryption, even perfect encryption with one-time pads, unless one also hides the traffic patterns by constantly sending encrypted meaningless messages ("link encryption"), which wastes too much bandwidth for most applications.

Interestingly, not even link encryption with one-time pads on authentic channels (to avoid destruction of messages) is as secure as this "nicest" ideal system in the synchronous model: In the real system, a message between two honest participants is first switched by the sender's machine, then the recipient's machine. The ideal system must represent the same delay. However, an adversary to whom a message is addressed (i.e., he can decrypt it), can obtain it without the delay of his own machine and send an answer back in the same way. Thus the answer arrives two rounds earlier than it would in the nicest ideal system. This is one of the subtleties of the synchronous model mentioned in Section 2.1 that might be abstracted away in an asynchronous model, but indicates a possible problem with timing in real life. (One cannot

[9] It should also become clear in the example that this is *not* the place for cryptographic vulnerabilities, neither unavoidable ones like the very small error probabilities, nor those whose absence should be proved.

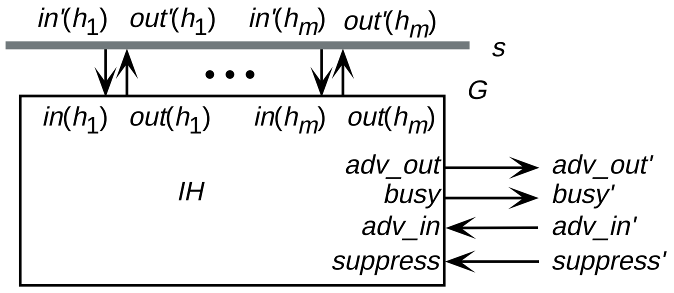

Fig. 5. A structure of the ideal system for secure message transmission.

assume that the honest *users* have no real-time clocks; hence such a speed difference may lead to different user behaviour in the real world.)

We therefore use the following ideal system as the specification, see Figure 5.

Definition 3.1 (Ideal System for Secure Message Transmission) Let a set Msg of messages of bounded length be given, a number $rnds$ of rounds, and a number n of participants, and let $\mathcal{N} = \{1, \ldots, n\}$. Then an ideal system Sys_2^{MT} for message transmission is given as follows: For any set $\mathcal{H} \subseteq \mathcal{N}$ of honest participants there is one structure with only one machine, called ideal host IH (we omit an index $\mathcal{H}$). It has one pair of free in- and output ports per honest participant, and these form the set s of specified ports.[10] Additionally, it has four ports for the adversary. adv_in' and adv_out' are for sending and receiving messages, while $busy'$ and $suppress'$ correspond to the imperfection of the traffic: In $busy'$, the adversary gets one bit of information for each message is transit, and with $suppress'$ he can suppress it (modeling that cryptography cannot guarantee availability).

We model that in each round $i > 0$, each participant can send one message to each other participant. Hence in each round i, the ideal host IH accepts inputs of the following form, where $\mathcal{A} = \mathcal{N} - \mathcal{H}$:

$$\begin{aligned} in_i &: \mathcal{H} \times \mathcal{N} \to Msg, \\ adv\_in_i &: \mathcal{A} \times \mathcal{H} \to Msg, \\ suppress_i &: \mathcal{H} \times \mathcal{H} \to \{0, 1\}. \end{aligned}$$

Here $in_i(h, \cdot)$ is the vector of inputs at port $in(h)$, and intuitively $in_i(h, l)$ is the message user h wants to send to user l in this round. We use ϵ for "no message". The switching function is, for $h, l \in \mathcal{H}$ and $a \in \mathcal{A}$:

[10] Having free ports in the graph G, instead of simply leaving some ports of IH free, may seem strange here. But in a few examples, the structures specifically offer multicast connections to their environment.

$$
\begin{aligned}
busy_i(h,l) &:= \begin{cases} 0 & \text{if } in_i(h,l) = \epsilon \\ 1 & \text{else;} \end{cases} \\
out_i(h,l) &:= \begin{cases} \epsilon & \text{if } suppress_i(h,l) = 1 \\ in_{i-1}(h,l) & \text{else;} \end{cases} \\
out_i(a,l) &:= adv\_in_i(a,l); \\
adv\_out_i(h,a) &:= in_i(h,a).
\end{aligned}
$$

The matrices $busy_i$ and adv_out_i are output at the corresponding adversary ports, and each vector $out_i(\cdot,l)$ at port $out(l)$.

Note how normal messages have a delay of one round, while messages to and from the adversary have not, and that "busy" and "suppress" are just 1-bit signals for each message, i.e., as abstract as possible.

We can now see why we have to distinguish the specified ports s, and to forbid H to use ports from $free(G_2, M_2) - s_2$ in the simulatability definition: We do not want the real system to have ports *busy* and *suppress*. Instead, it will have normal channels where encrypted messages will pass. Hence, while we require that the events at the user ports $in(h)$ and $out(h)$ are simulated indistinguishably, we do not require this for the ports *busy* and *suppress*. This becomes even more important in a larger example like our certified mail example, where, e.g., one "busy" signal per protocol run is enough, i.e., we do not even need one per message. This simplifies the abstract specification and enables it to be protocol-independent. (Recall the alternative in [22] mentioned in Section 2.1.)

Scheme 1 (Real message transmission, sketch) In the actual system Sys_1^{MT} that we prove to be as secure as the ideal system, there is one machine M_l per intended user l, and all are connected by point-to-point channels. The actual structures (for any subset $\mathcal{H}$ of honest participants) are derived with a standard trust model with insecure channels, except that authentic channels are needed for key exchange (only used in Round 0): Only the machines M_h with $h \in \mathcal{H}$ are present in M, the insecure channels are split into one channel to and one from the adversary, and the authentic channels get the adversary as an additional recipient.

In each round $i > 0$ (after initial key generation and exchange), each correct machine M_h transforms its user inputs $in_i(h,l)$ into network messages of the following form:

$$
netw_i(h,l) \leftarrow \begin{cases} E_l(h, sign_h(in_i(h,l), i, l)) & \text{if } in_i(h,l) \neq \epsilon, \\ \epsilon & \text{else.} \end{cases}
$$

E denotes encryption as before, and *sign* signing, including the message in clear. The comma denotes tuple composition, not concatenation, and the implementation must guarantee unambiguous decomposition.

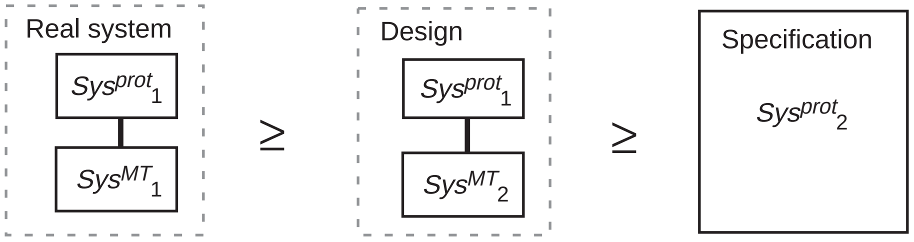

Fig. 6. Simplest form of the composition theorem. The symbol "≥" stands for "at least as secure as".

In each round $i > 0$, M_h also decomposes messages $netw'_{i-1}(l, h)$ from the network as follows. Let

$$(from_{i-1}(l,h), sig_{i-1}(l,h)) \leftarrow D_h(netw'_{i-1}(l,h))$$

or, if the decryption or decomposition fails, let both components be ϵ. Then

$$out_i(l,h) := \begin{cases} m & \text{if } from_{i-1}(l,h) = l \wedge test_l(sig_{i-1}(l,h)) = (m, i-1, h) \\ \epsilon & \text{else.} \end{cases}$$

We can prove that this system is as secure as the ideal system Sys_2^{MT} if the encryption system and the signature system are secure under their normal cryptographic definitions, in particular the one shown in Section 1.2. Omitting any parameter from the network messages, e.g., the outer h, makes the protocol insecure. But of course, quite different protocols may be as secure as the same ideal system.

4 Composition

One of the main advantages of our model is a composition theorem. We will briefly sketch what we mean by this and its relation to formal methods. [11]

Consider the example of secure message transmission. The ideal system Sys_2^{MT} contains no probabilism at all, to say nothing of encryption operations etc. It is a very simple, non-distributed system. If a larger protocol Sys_1^{prot} makes use of secure message transmission, we therefore want to design it assuming that the message transmission is done by the ideal system Sys_2^{MT}. If the protocol uses cryptography in no other form, we can therefore design it without any probabilism etc., and thus hopefully in a simple language and even with the support of standard tools. Hence Sys_2^{MT} plays the role of an abstract primitive in Figure 2.

Now the larger protocol will also have some specification; we call this Sys_2^{prot}. It corresponds to the abstract goals in Figure 2. This is shown in Figure 6. Although we also wrote the right part with "≥", denoting "at least

[11] We have a proof, but at the time of this writing sketchy, while the examples sketched in Section 3 are fully worked out in [32,33].

as secure as", we would hope that the proof does not involve any arguments with probabilities etc. because both systems are deterministic; hence it should be accessible to formal proof methods.

What the composition theorem gives us is the left part: The real higher protocol using the real secure message transmission system must be as secure as its design that used the specification of the message transmission system. Of course, the theorem does this generally, not only for message transmission on the lower layer. Moreover, "$\geq$" is indeed transitive, as suggested in the figure.

The theorem can be extended to the composition of several systems, and parallel composition is a special case.

Another theorem that we have actually proven is a kind of right part of Figure 2 for the case where the abstract goals are not a specification by an ideal system, but by linear-time logical expressions over the specified ports: If a system Sys_2 fulfils such requirements, and Sys_1 is as secure as Sys_2 (with respect to a mapping that keeps the specified ports constant) then so does Sys_1, in a well-defined notion of computationally fulfilling linear-time formulas.

5 Relation to Formal Methods

The main relations to formal methods were already sketched in Section 4: Our current specifications of abstract ideal hosts are mathematically rigorous, but informal. However, they clearly lend themselves to a range of normal specification techniques. Then the composition theorem shows ways how to prove larger systems secure without much special attention to the specific cryptographic semantics.

Our abstractions in the two large examples are on a rather high layer, e.g., we did not present an abstract model of encryption, but immediately of secure message transmission. It will certainly be interesting to see how much lower one can get or to what extent current cryptographic protocols have to be and can be redesigned to use only the abstract primitives that do have a cryptographic semantics. Here combinations with some of the less abstract formalizations mentioned above will certainly be interesting.

References

[1] M. Abadi, *Protection in Programming-Language Translations*, 25th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 1443, Springer-Verlag, Berlin 1998, 868–883

[2] M. Abadi, A. D. Gordon, *A Calculus for Cryptographic Protocols: The Spi Calculus*, 4th Conference on Computer and Communications Security, ACM,

New York 1997, 36–47

[3] M. Abadi, M. R. Tuttle, *A Semantics for a Logic of Authentication*, 10th Symposium on Principles of Distributed Computing (PODC), ACM, New York 1991, 201–216

[4] D. Beaver, *Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority*, Journal of Cryptology 4/2 (1991) 75–122

[5] M. Bellare, R. Canetti, H. Krawczyk, *A modular approach to the design and analysis of authentication and key exchange protocols*, 13th Symposium on Theory of Computing (STOC), ACM, New York 1998, 419–428

[6] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 26–45

[7] M. Blum, S. Micali, *How To Generate Cryptographically Strong Sequences Of Pseudo Random Bits*, 23rd Symposium on Foundations of Computer Science (FOCS), IEEE, 1982, 112–117

[8] R. Canetti, *Studies in Secure Multiparty Computation and Applications*, Thesis, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, June 1995, revised March 1996

[9] R. Canetti, *Security and Composition of Multi-party Cryptographic Protocols*, Theory of Cryptography Library 98-18, June 1998, last revision August 1999, http://philby.ucsd.edu/cryptolib/

[10] R. Canetti, S. Goldwasser *An Efficient Threshold Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack*, Eurocrypt '99, LNCS 1592, Springer-Verlag, Berlin 1999, 90–106

[11] R. Cramer, V. Shoup, *A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack*, Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 13–25

[12] D. Dolev, A. C. Yao, *On the Security of Public Key Protocols*, IEEE Transactions on Information Theory 29/2 (1983) 198–208

[13] S. Even, O. Goldreich, A. Shamir, *On the Security of Ping-Pong Protocols when Implemented using the RSA*, Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 58–72

[14] R. Gennaro, S. Micali, *Verifiable Secret Sharing as Secure Computation*, Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 168–182

[15] O. Goldreich, *Secure Multi-Party Computation*, Working Draft, Version 1.1, September 21, 1998, available from http://www.wisdom.weizmann.ac.il/users/oded/pp.htm

[16] O. Goldreich, S. Micali, A. Wigderson, *How to play any mental game—or—a completeness theorem for protocols with honest majority*, 19th Symposium on Theory of Computing (STOC), ACM, New York 1987, 218–229

[17] S. Goldwasser, L. Levin, *Fair Computation of General Functions in Presence of Immoral Majority*, Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 77–93

[18] S. Goldwasser, S. Micali, C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*, SIAM Journal on Computing 18/1 (1989) 186–207

[19] M. Hirt, U. Maurer, *Player Simulation and General Adversary Structures in Perfect Multi-Party Computation*, Swiss Federal Institute of Technology (ETH), Zurich, Dec. 1997, invited to a special issue of Journal of Cryptology, available from http://www.inf.ethz.ch/personal/hirt/publications/journal.ps.gz

[20] J. Jacobs, *Security specifications*, 1988 Symposium on Security and Privacy, IEEE, Washington 1988, 14–23

[21] E. S. Lee, B. W. Thomson, P. I. P. Boulton, R. E. Soper, *Composable Trusted Systems*, CSRI, University of Toronto, Report CSRI-272, May 1992

[22] P. Lincoln, J. Mitchell, M. Mitchell, A. Scedrov, *A Probabilistic Poly-Time Framework for Protocol Analysis*, 5th Conference on Computer and Communications Security, ACM, New York 1998, 112–121

[23] P. Lincoln, J. Mitchell, M. Mitchell, A. Scedrov, *Probabilistic Polynomial-Time Equivalence and Security Analysis*, Formal Methods 1999, available at ftp://theory.stanford.edu/pub/jcm/papers/fm-99.ps

[24] G. Lowe, *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*, Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS 1055, Springer-Verlag, Berlin 1996, 147–166

[25] C. Meadows, *Using Narrowing in the Analysis of Key Management Protocols*, 1989 Symposium on Security and Privacy, IEEE, Washington 1989, 138–147

[26] S. Micali, C. Rackoff, B. Sloan, *The Notion of Security for Probabilistic Cryptosystems*, SIAM Journal on Computing 17/2 (1988) 412–426

[27] S. Micali, P. Rogaway, *Secure Computation*, Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 392–404

[28] J. K. Millen, *The Interrogator: A Tool for Cryptographic Protocol Security*, 1984 IEEE Symposium on Security and Privacy, IEEE, Washington 1984, 134–141

[29] B. Pfitzmann, *Sorting Out Signature Schemes*, 1st Conference on Computer and Communications Security, ACM, New York 1993, 74–85

[30] B. Pfitzmann, *Cryptographic Semantics of Formal Specifications*, presented at Colloquium on Formal Methods and Security, Isaac Newton Institute, University of Cambridge, April 23, 1996

[31] B. Pfitzmann, *Digital Signature Schemes—General Framework and Fail-Stop Signatures*, LNCS 1100, Springer-Verlag, Berlin 1996

[32] B. Pfitzmann, M. Schunter, M. Waidner, *Secure Reactive Systems*, IBM Research Report RZ 3206 (#93252), IBM Research Division, Zürich, Feb. 2000

[33] B. Pfitzmann, M. Schunter, M. Waidner, *Provably Secure Certified Mail*, IBM Research Report RZ 3207 (#93253), IBM Research Division, Zürich, Feb. 2000

[34] B. Pfitzmann, M. Waidner, *A General Framework for Formal Notions of "Secure" System*, Hildesheimer Informatik-Berichte 11/94, Universität Hildesheim, April 1994, available at http://www.semper.org/sirene/lit/abstr94.html#PfWa_94

[35] B. Pfitzmann, M. Waidner, *Extensions to Multi-Party Computations*, The 1998 Weizmann Workshop on Cryptography, June 16-18th, Rehovot, Israel, slides available at http://www.semper.org/sirene/lit/abstr98.html#PfWa3_98

[36] A. W. Roscoe, *Modelling and Verifying Key-Exchange Protocols Using CSP and FDR*, 8th Computer Security Foundations Workshop, IEEE, Los Alamitos 1995, 98–107

[37] S. Schneider, A. Sidiropoulos, *CSP and Anonymity*, 4th European Symposium on Research in Computer Security (ESORICS), LNCS 1146, Springer-Verlag, Berlin 1996, 198–218

[38] V. Shoup, *On Formal Models for Secure Key Exchange*, IBM Research Report RZ 3076 (##93122), IBM Research Division, Zürich, November 1998, also Theory of Cryptography Library 99-12, last revised November 1999, http://philby.ucsd.edu/cryptolib/

[39] P. Syverson, C. Meadows, *A Logical Language for Specifying Cryptographic Protocol Requirements*, 1993 Symposium on Research in Security and Privacy, IEEE, Los Alamitos 1993, 165–177

[40] M. J. Toussaint, *A New Method for Analyzing the Security of Cryptographic Protocols*, IEEE Journal on Selected Areas in Communications 11/5 (1993) 702–714

[41] G. Wedel, V. Kessler, *Formal Semantics for Authentication Logics*, 4th European Symposium on Research in Computer Security (ESORICS), LNCS 1146, Springer-Verlag, Berlin 1996, 219–241

[42] A. C. Yao, *Protocols for Secure Computations*, 23rd Symposium on Foundations of Computer Science (FOCS), IEEE, 1982, 160–164

[43] A. C. Yao, *Theory and Applications of Trapdoor Functions*, 23rd Symposium on Foundations of Computer Science (FOCS), IEEE, 1982, 80–91