Sorting Out Signature Schemes

Birgit Pfitzmann

Universität Hildesheim, Institut für Informatik,
Marienburger Platz 22, D-31141 Hildesheim, Germany
pfitzb@informatik.uni-hildesheim.de

Abstract. Digital signature schemes are a fundamental tool for secure distributed systems. It is important to have a formal notion of what a secure digital signature scheme is, so that there is a clear interface between designers and users of such schemes. A definition that seemed final was given by Goldwasser, Micali, and Rivest in 1988. Since then, however, several signature schemes with new security properties have been presented, which are not covered by this definition. Hence the new properties were not defined as additions, but each new type of scheme needed a new definition from scratch. This was unsatisfactory.

This paper presents an overview of a general definition of digital signature schemes that covers all these schemes, and hopefully all that might be invented in future. Additional properties of special types of schemes are then presented in an orthogonal way, so that existing schemes can be classified systematically. It turns out that signature schemes are best defined by a separation of service, structure, and degree of security. For the service specification, we use temporal logic. Several parts of such a definition can easily be reused for general definitions of other classes of cryptologic schemes.

1 Introduction

The purpose of digital signature schemes is to provide message authentication such that disputes about the authenticity of a message can be settled by third parties, such as courts. Digital signature schemes are therefore needed whenever messages of legal relevance are exchanged over communication networks, and they are also an important tool in distributed secure systems. The property that third parties can settle conflicts between the sender and the recipient is also called non-repudiation; it distinguishes signature schemes from simpler symmetric authentication schemes.

As with all security-critical systems, it is important to have a formal definition of what a digital signature scheme is, so that the design and the application of such schemes can be separated. Applications should work with any digital signature scheme that fulfils the definition, without caring about internal details, and designers should know what properties are expected from their schemes.

1.1 Definitions of Ordinary Digital Signatures

Although signature schemes are rather "small" schemes, e.g., in comparison with secure operating systems or, within cryptology, general multi-party computation protocols, agreeing on a definition has not been trivial. The first informal definition by Diffie and Hellman [DH76]

A preliminary version of this paper was presented at the 1st ACM Conference on Computer and Communications Security, Fairfax, acm press 1993, 74-85.

saw signature schemes as a form of encryption schemes used in an inverse way, but nowadays many schemes, e.g., the proposed NIST digital signature standard, are not of this form [DSS92]. Moreover, it was originally overlooked that attackers might try to compute new signatures from given signatures, in particular with RSA [RSA78, D84]. Hence it was very satisfactory that a seemingly final definition was given in [GMR88]. It allows the scheme to consist of an arbitrary key generation algorithm *gen*, an arbitrary signing algorithm *sign*, and an arbitrary test algorithm *test*, where *gen* and *sign* are probabilistic and *test* is usually required to be deterministic; see Figure 1.

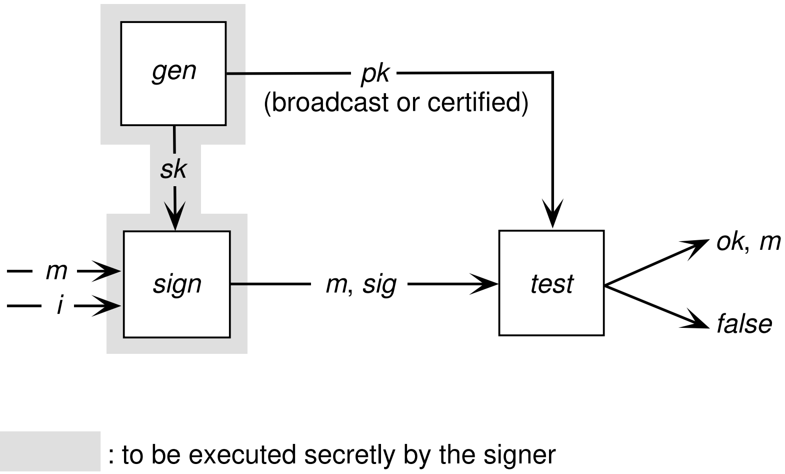

Figure 1 Components of a signature scheme in the GMR definition and their use. The algorithms *gen* and *sign* are probabilistic. (*sk*, *pk*) is a pair of a secret and a public key, *m* a message, *sig* a signature, and *i* represents the number of messages signed so far.

Moreover, a strong security definition covering all conceivable behaviors of an attacker was given: The attacker may first repeatedly ask the signer to sign a message (adaptive chosen-message attack), and then the attacker is said to be successful if he can produce any additional signed message (existential forgery).

The GMR definition is important in practice even though most schemes in actual use cannot be proved secure with respect to it: They are all designed so that one can at least hope that they fulfil it. For instance, one does not use "pure" RSA in practice, but includes hash functions or redundancy to counter the above-mentioned attacks; see the ISO standardization activities or the proposed NIST digital signature standard [GQ91, ISO91, DSS92]. Note that hash functions or redundancy have to be considered parts of the signature scheme if one wants to provide a common interface to applications. Moreover, note that an unrestricted chosen-message attack, where the signer signs anything the attacker tells her to, seems a bit excessive, but no restricted form is known where one is fairly sure to be on the safe side: It is unlikely that a signer will sign *any* message, but *what* message could one be sure she would not sign? Hence one really takes measures against the general attack in practice.

1.2 Constructions of Ordinary Digital Signature Schemes

It would exceed the scope of this paper to give an overview of the constructions of ordinary digital schemes.

However, we can briefly survey the schemes whose security has been proved under reasonable cryptographic assumptions. The first one was presented in [GMR88], with efficiency improvements in [G87]. For an implementation, see [FP91]. The necessary assumptions were successively weakened in [BM92, NY89, R90]. Recently, efforts have been made to improve the efficiency in [BC93] (with an incomplete proof) and [DN94, CD95]. Note, however, that fail-stop signature schemes (see below) also yield provably secure ordinary digital signature schemes, see [PW90, PP95], and thus the first efficient provably secure scheme based on discrete logarithms was [HP93].

1.3 New Schemes

Just when this satisfactory state with a generally accepted definition was reached, new types of signature schemes with new security properties came up.

- With *fail-stop* signature schemes, a signer who has been cheated with a forged signature can prove that it was a forgery [PW90, BPW91, Pf91, HP93, HPP93, DPP94, PP95].
- With so-called *undeniable* signature schemes [CA90, C91, BCDP91], the recipient of a signature cannot show it around, e.g., to his friends, without the signer's help. Because of this property, these schemes might rather be called "invisible", as not being deniable is common to all types of signatures. The invisibility property is useful if it should be private what messages the users sign. However, if a signer is forced to either acknowledge or deny a signature, e.g., in court, she cannot falsely deny her real signatures.
- There are even *unconditionally secure* signature schemes, where even an attacker with unlimited computing power cannot produce forgeries better than by using a random number and hoping that it is the signature [CR91, PW92]. However, unconditionally secure signature schemes seem to be impractical, in contrast to fail-stop and undeniable ones.

All these new schemes exist in several *variants* with different properties. For instance, fail-stop signature schemes can be more efficient if all the signatures of one signer have the same recipient. Such a scheme could be applied in a payment system where clients only sign money transfer orders for their bank. Another example is that the original unconditionally secure schemes did not enable recipients to transfer signatures from one to another, which one tacitly takes for granted with most other digital signature schemes, but one can reintroduce this property in a limited way. A third example is undeniable signature schemes where signers can unconditionally deny forged signatures, which is a slightly restricted fail-stop property [CHP92] (see the definition of dual security in Section 6.2).

This situation alone might call for some classification. However, it would not be too bad if all these schemes were signature schemes in the sense of the GMR definition with some additional properties. However, none of them is, because they all differ somehow in their *structure*, i.e., in the number of algorithms and their parameters, for instance:

- In fail-stop signature schemes, the recipients or a center trusted by them must take part in key generation, so that they can be protected from the signer falsely proving her own signatures to be forgeries.
- Undeniable signature schemes do not have a test algorithm in the usual sense, but an interactive protocol, usually a zero-knowledge proof, where the signer proves the correctness of the signature to the recipient.
- In both fail-stop and undeniable signature schemes, a third party cannot simply test a signature presented by a recipient, but must also ask the signer, so that the signer has a chance to produce a proof of forgery or to carry out a disavowal protocol, respectively.
- In unconditionally secure signature schemes, there is no public key in the usual sense; instead, everybody obtains a different test key in a complicated key generation protocol.

Not even all the signature schemes that are supposed to be ordinary, i.e., that have no special new features, are covered by the GMR definition: Sometimes, signing is not only an algorithm with the inputs sk, m, and i, but some memory is needed between executions, e.g., in Schnorr's scheme with preprocessing [S91]. In another example, there is a trusted public random value in addition to the public keys [BC93].

1.4 Goals of Definition and Classification

So what does one do with all these new types of schemes?

Even if each one had a new definition, which some do, but most variants don't, the situation would be unsatisfactory: First, if they are all called signature schemes, a fact that nobody seems to challenge, they ought to have something in common. Secondly, new definitions in cryptology are just as error-prone as new schemes; hence one should have as few definitions as possible and evaluate them carefully.

Hence the properties common to all signature schemes should be stated clearly once and for all. Special properties should be formulated as additions to this general definition. Applications then know what to expect from all signature schemes, or can list the additional properties they need. Such a definition is given in this paper.

1.5 Scope

The scope of signature schemes actually defined here is schemes that could be used in the place of handwritten signatures in law, i.e., where each signer has complete control over what she signs. This excludes *blind* and *group* signature schemes, e.g., [C85] and [DF92, CH91] — their services are too different for a nice joint definition. For similar reasons, we have excluded schemes that need additional parties, e.g., *identity-based* schemes [S85] and schemes with specific helpers for the signature verification [Pe91, C95]. The general parts of the definition and classification, however, could be applied to those schemes, too, and specifying the exact services of such schemes in our framework would be very useful future work.

2 The Main Ideas

Intuitively, what all signature schemes have in common is exactly what was written in the very first sentence: They provide message authentication such that disputes about the authenticity of messages can be settled by third parties. However, as sketched above, they have different *degrees of security*, such as fail-stop, and their *structure* varies widely. All existing definitions of digital signature schemes specify what disputable message authentication means only in terms of one given structure and with one particular degree of security. This is why the whole definition had to be changed whenever the structure was modified or a different degree of security was introduced.

The basic idea to obtain a general definition is therefore to separate the following three aspects:

- *Service*: Here we define the minimal service of all signature schemes in an ideal way; ideal means independent of the structure and the degree of security. We do this in a normal specification language, i.e., one which is not specific to cryptology. We chose temporal logic, but other languages with the same expressive power could be used as well.
- *Structure:* Here we define a very general structure common to all signature schemes.
- *Degree of security*: Here we define what it means that *any* scheme provides *any* service with a certain degree of security. In other words, we define a specific cryptologic semantics for the normal specification language.

Given such a general definition, one can *classify* particular types of signature schemes according to

- additional service properties,
- special (simpler) cases of the structure, and
- the actual degree of security.

An additional advantage of such a modular definition is that only the service is specific to signature schemes, whereas the structural aspects and the degrees of security can be reused with other schemes that still lack general definitions, e.g., digital payment schemes. Note, however, that only integrity and availability requirements can be expressed in temporal logic, not privacy. For an overview of how this approach can be extended to include privacy, see [PW94].

The service, both the minimal requirements used to define the notion of a signature scheme and additional properties, is considered in Section 4. Sections 5 and 6 treat structure and degree of security. The parts that are common to much more general schemes than signature schemes can only be sketched here.

3 Related Work

There are two somewhat related well-known approaches at general and abstract definitions of cryptologic schemes: definitions of secure multi-party computation protocols from theoretical cryptology and logics of authentication, in particular BAN logic, from the design of protocols using cryptologic primitives. For interested readers, the relations and differences to these approaches are now discussed. No knowledge of these approaches is assumed in the later sections.

3.1 Relation to Secure Multi-Party Computation Protocols

Deriving a cryptologic definition from an ideal service specification is new with signature schemes, but it is known with multi-party function computation, starting with [Y82]. Overviews can be found in [B91, MR91, FGY92]. However, existing definitions were not general enough to be used for signature schemes. This is no criticism — they go much further in other directions, in particular definitions of privacy. The following new features were needed:

- *Logic specification.* The services provided by different types of signature schemes are similar, but not identical. In multi-party function computation, however, the specification is given by exactly one ideal trusted host, which has to be simulated very closely by the actual distributed cryptologic protocol. Here, a specification expressing individual requirements is used instead. Hence we can formulate some minimal requirements that have to be fulfilled by all signature schemes, and add further requirements to specify particular types of schemes. Actually, not even individual types of signature schemes, e.g., the ordinary ones according to the GMR definition, simulate one ideal trusted host.
- *Reactive systems.* Current formal definitions of multi-party protocols only treat the computation of functions. Each user of such a system makes one initial input and obtains one output at the end. Signature schemes, however, yield reactive systems: They can go on (polynomially) forever, and at any time users can decide what messages they want to sign or what disputes to carry out. An informal sketch of more general games exists in [GMW87]; if they were formalized, they might be nearer to signature schemes in this respect. In [GM95], a specific 2-phase protocol is formally defined by requiring each phase to securely evaluate a certain function. However, this makes the internal state between the two stages an explicit output, whereas we want to abstract from the internal state.
- *Varying connections.* Usual multi-party protocols assume fixed connections throughout a protocol run, usually reliable point-to-point channels between each pair of participants and often reliable broadcast channels. With signature schemes, the connections and their security can vary as a function of time: Neither do the participants, which may be hand-held computers in pockets, have so many dedicated channels, nor are the channels secure, except for the broadcast channel during initialization — this is part of the reason to use signatures.

3.2 Relation to BAN Logic

Logic specification of signature schemes may sound rather similar to the logics of authentication from [BAN89] and many successive papers. However, there are several differences in the purpose and consequently in the approach. This is natural, because these logics were primarily designed to find errors in key exchange protocols with symmetric cryptosystems, which they do successfully.

- Logics of authentication are used to specify and prove protocols *using* cryptologic primitives, whereas here, the signature schemes themselves are to be specified.
- Logics of authentication assume very simple cryptologic primitives; hence they deal with individual algorithms and keys of those schemes. Hence a new type of logic would be needed for each new type of signature scheme, precisely as with current cryptologic definitions. Usually, e.g., in [LABW92], integrity is still assumed to be provided by a variant of an encryption scheme, a structure that most signature schemes do not have. A more general abstraction would use the algorithms from the GMR definition, i.e., Figure 1.
- If logics of authentication have a formal semantics, not the real cryptologic schemes are used, but algebraic abstractions [DY81, AT91]. Hence, if one finds an error with the logic, there was an error in the real protocol, but if one proves the algebraic abstraction to be secure, one is not sure that the real protocol is secure, too. Here, the real signature schemes are the semantics.

An approach more similar to ours was taken in [SM93], where requirements on the schemes usually treated with BAN logic were also formulated independently of the structure and in temporal logic. (The semantics is algebraic abstractions as above.) The general type of requirements is very similar, except that [SM93] has no explicit interface as ours below. Introducing one might be helpful there, too, because intuition seems to be needed to map, e.g., an accept statement from the requirements to something in the actual protocol, whereas here, it *is* the interface output of the protocol. We also found a separation of system entities and users helpful because entities have prescribed programs and users do not.

For an overview of other related approaches, see [M95].

In future, the approaches might meet in logics for the design of complicated protocols, but using more general primitives and a real cryptologic semantics.

4 Service Specification of Signature Schemes

We now give a logic specification of the service of all digital signature schemes.

4.1 What is Specified?

To achieve the desired independence of structural differences, we regard the whole signature scheme as one system whose behavior at the *interface* to its users is specified, see Figure 2. (More precisely, this "system" is an instantiation of the "scheme", see Section 5.1.) Specifying interface behaviour is normal in the field of distributed systems, e.g., [L83], but not in cryptology.

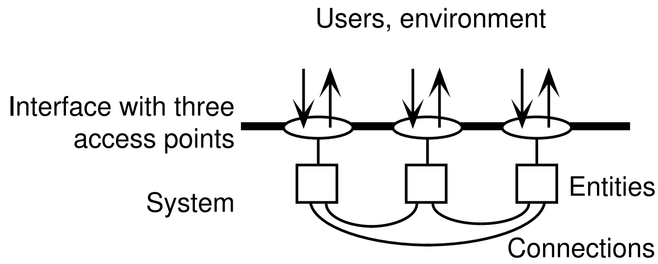

Figure 2 A system and its interface. The service to be specified is the set of permitted sequences of interface events.

In our case, the entities correspond to processes running on the behalf of individual users. For instance, if the signature scheme is implemented in hand-held computers used by individuals, an entity roughly corresponds to the processes on one such computer. The users of the signature scheme are in the environment, hence they are not modeled explicitly, but intuitively one can assume that each access point is owned by a particular user. This becomes more important when degrees of security are modeled below and users are regarded as mutually distrusting.

4.2 Overview of the Interface Events

So what are the interface events of a signature scheme, i.e., what are the users concerned with, and not only the entities inside the system? Informally, the users need the system to perform at least three types of so-called transactions, each consisting of a few related interface events at several access points.

- **Authentication.** Any two users, called a signer and a recipient, can authenticate a message. The signer inputs which *message* is authenticated. The recipient has to know the message, too, and the *identity* of the signer who is supposedly authenticating it. At the end, the system must give the recipient a Boolean output indicating whether the message was *accepted* as authenticated or not.
- **Disputes.** In a dispute, the third party trying to settle it and the recipient who claims to have the authenticated message take part, and sometimes the supposed signer, too. As in authentication, the users have to input the *message* that is disputed and the *identity* of the signer who is supposed to have authenticated it, and the third party obtains a Boolean output indicating whether the message was *accepted* as authenticated or not.
- **Initialization.** For technical reasons, initialization is needed. It corresponds to key generation and distribution inside the system, but the users only need to know the *identity* of the (future) signer for whom this is happening and whether initialization was *successful*. All users should take part in initialization.

We call these the *minimal transactions* of a signature scheme, because every scheme must offer them. Specific schemes may offer more.

Note that the interface events only deal with messages, identities, and Boolean values denoting if something was accepted or not — signatures and keys are hidden *inside* the system.

4.3 Interface Events in Detail

More formally, the domain of the interface events are defined as follows: Each event is represented as a triple

$$(identity, direction, value),$$

where *identity* denotes the access point where the event occurs, $direction \in \{in, out\}$ denotes whether it is an input or an output event, and *value* is the value passed through the access point. The domain of *identity* is a set $Id = Id_S \cup Id_R \cup Id_T$, i.e., separate access points are used when acting as a signer, recipient, or third party.

The domains of *value* are a bit more complicated than sketched in Section 4.2. For the minimal transactions, they are shown in Table 1. The individual parameters of the values have the following meaning:

- Each input value starts with a string, such as *'sign'*, that characterizes the type of action wanted.
- m denotes a message from a message space M.
- id_S denotes an identity of a signer, i.e., an element of Id_S.
- ids_R is a description of a set of recipients. The domain of such a parameter may be an arbitrary subset of the power set of Id_R. These parameters are needed because inside some signature schemes, signing depends on the identities of the intended recipients. The empty string ε is used as the description of the complete set, i.e., for "no restriction" — this is the standard case.

		Signer	Recipient(s)	Third party/ies
Authen-tication	In:	***'sign'***, $\boldsymbol{m}$, ids_R	***'test'***, $\boldsymbol{m}$, $\boldsymbol{id_S}$	—
	Out:	*eot*	$\boldsymbol{acc}$ $\in$ {TRUE, FALSE}	—
Dispute	In:	*'disavow'*, m, ids_R	***'show'***, $\boldsymbol{m}$, $\boldsymbol{id_S}$	***'decide'***, $\boldsymbol{m}$, $\boldsymbol{id_S}$, ids_R
	Out:	*eot*	*eot*	$\boldsymbol{acc}$ $\in Dispute\_results$ $\supseteq$ {TRUE, FALSE}
Initial-ization	In:	***'init'***, ids_{in}, N	***'init'***, $\boldsymbol{id_S}$, ids_{in}, N	***'init'***, $\boldsymbol{id_S}$, ids_{in}, N
	Out:	acc, ids_{out}	acc, ids_{out}	acc, ids_{out}

Table 1 Interface events and their parameters. Those that would also be needed if only ordinary digital signature schemes were specified are in bold face.

- The two output values **acc** in bold face denote if a message was accepted as authenticated. Disputes in some schemes may have results other than Boolean, hence a domain *Dispute_results* $\supseteq$ {TRUE, FALSE} is permitted. The two most important cases are
  - *acc* = *'broken'* in fail-stop signature schemes means that a valid proof of forgery has been produced in the system, and thus the users can no longer rely on this system.
  - *acc* = *'not_cooperated'* in undeniable signature schemes means that the signer refuses to cooperate and no decision can be reached in the system. The users outside the system must decide what to do with non-cooperating signers under which circumstances.

  In initialization, *acc* denotes whether this initialization was successful at all, i.e., whether it is now possible to authenticate messages under id_S.
- The outputs *eot* are usually only needed to indicate that the transaction has ended.
- The parameters ids_{in} in initialization denote which access points, and thus users, are supposed to take part, and ids_{out} denotes which of them actually took part successfully. This is needed to cover schemes with interactive key generation where disrupters have to be excluded.
- N is a message bound from a set *Bounds* $\subseteq \mathbb{N} \cup \{\infty\}$, i.e., the number of messages the signer can authenticate at this access point after this initialization.

Note that the signer does not input her own identity during authentication — it is implicit in the access point she uses, usually in the form of a secret key stored in the entity serving the access point. Restricting the access to the access point, and thus the usage of the key, is not the business of the signature scheme, but, e.g., of password protection. Moreover, a service where one could input any identity at any access point would be insecure, as anyone could input the identity of anyone else. Similarly, the signer does not input her identity in initialization. In that case, it is usually present in the entity in the form of access to a broadcast channel under this identity, so that the generated key can be distributed under the identity.

4.4 Specification and System Parameters

The domains of interface events had some parameters, such as a message space M. Hence there is a whole family of specifications $Spec_{pars}$, where *pars* denotes the tuple of these parameters. They will be treated in two different ways:

- **Specification parameters.** Almost all parameters can be fixed by the designer of a concrete signature scheme. This makes the definition very general.

  Hence there may be signature schemes for any message space M, say, with $M \subseteq \{0, 1\}^+$, any set *Bounds* $\subseteq \mathbb{N} \cup \{\infty\}$ of message bounds, any countable set *Dispute_results* $\supseteq$ {TRUE, FALSE}, and a few more parameters. In particular, there is a predicate $suitable(Id_R, ids_{out}, ids_R)$ that describes which sets of identities, ids_R, are suitable inputs by the third party in a dispute if Id_R is the set of all identities and ids_{out} that of successful participants in the corresponding initialization.

  The tuple of all specification parameters is called *spec_pars*.

- **System parameters.** The exception is that any signature scheme will have to work for almost arbitrary sets Id_S, Id_R, and Id_T of identities; see Section 5.

4.5 Logic Requirements

The requirements on signature schemes specify what outputs the systems have to produce, depending on previous interface events. Hence they are formulas that describe some sequences of interface events as permitted and others as forbidden. Recall that these requirements are made in an ideal way; error probabilities etc. are handled once and for all in the degrees of security.

Note that we are lucky that all minimal requirements on signature schemes only specify what outputs may occur: There are neither privacy requirements nor probability distributions. Hence normal temporal logic or other formalisms for describing sets of sequences can be used. (Of course, there will be secret information inside the systems, but this is only the implementation.)

Each requirement is formulated from the point of view of some **interested parties**, formally a set of identities. It only involves events at the access points of these parties. In Section 6, such a requirement has to be fulfilled even if all the other parties are attacking and their entities deviate from the protocol.

4.5.1 Overview of the Minimal Requirements

First, an informal and simplified overview of the requirements that all signature schemes must fulfil is given.

- **Requirement of the recipient on disputes.** Once a recipient has accepted a certain message m as authenticated by a certain signer id_S, i.e., he made an input (*'test'*, m, id_S) and the output was acc = TRUE, the message should also be accepted by any third party in a dispute, i.e., the third party's output acc should be TRUE. To include schemes like undeniable signature schemes, it must be tolerated that the output is *'not_cooperated'* instead.

  The interested parties are the recipient and the third party who settles the dispute. Note that the whole requirement is formulated from their point of view: No event at the signer's access point is mentioned. Informally, only the recipient might be called "interested", but the requirement presupposes that the third party is honest, and it can only be fulfilled if the third party's entity is correct.

- **Requirement of the signer on disputes.** As long as a signer has not authenticated a certain message m, i.e., she has not input (*'sign'*, m, ids_R) for any ids_R, a third party should not accept that she did, i.e., the third party's output acc in a dispute should be FALSE, at least if the signer disavows the message in this dispute. The interested parties are the signer and the third party.

- **Effectiveness of authentication.** If a signer and a recipient make consistent inputs to start authentication, i.e., with the same message and identities, the message should be accepted, i.e., the recipient should obtain the output acc = TRUE.

- **Effectiveness and correctness of initialization.** If an initialization is started, and at least the future signer and one further participant are interested, it should end successfully, i.e., with acc = TRUE. Moreover, the outputs acc and ids_{out} denoting whether it worked at all, and with whom, should be consistent among any set of interested parties, and nobody who took part correctly should be missing from ids_{out}.

- **Availability of service.** Any authentication, dispute, or initialization should end within reasonable time, so that new inputs are accepted at the same access point. The interested party is only the owner of the access point.

One may miss **unforgeability** in this list, i.e., if a signer has not authenticated a certain message, a recipient should not accept that she did. However, one can show that this is a consequence of the other requirements, with one exception of only technical interest [P94].

The complete formalization of the requirements exceeds the scope of this paper. However, some general details and one example are presented.

4.5.2 Some Details

First, one has to decide to what extent honest users are assumed to *"behave reasonably"*. For the general definition, we want minimal requirements on the system, and thus strong preconditions about the users. In particular, the minimal requirements have preconditions that initialization with the correct interface events at the access points of the interested parties has been performed before any other transaction for the same identity id_S of a signer. More user-friendliness, e.g., error messages in other cases, can be added as additional service properties.

Secondly, varying *connections* have to be handled, see Section 3.1. For instance, the requirement of the recipient on disputes can only be fulfilled if the entities of the recipient and the third party are connected during the dispute, e.g., so that a signature can be forwarded inside the system, but no precondition about the connections during authentication is needed. Hence the necessity of connections has to be specified in the requirements. For this, a variable *connections* is introduced that contains, at each point in time, the current connections. It can be visualized as an additional access point where all the connections are controlled. For generality, each connection is expressed as a pair (*transaction_type*, *identity_set*). The mapping *channel_types* that says what channels, e.g., "reliable point-to-point" or "broadcast", are actually established between these identities in such a transaction is an additional specification parameter. Only point-to-point channels are permitted in authentication and disputes, whereas broadcast channels can be used in initialization.

Thirdly, a model of *time* is needed. At present, a simple synchronous model is used. In each round, inputs can be made at any number of access points, and then outputs at some access points may occur. This may be extended in future.

Finally, it is assumed for simplicity that users who carry out a *transaction* together start it synchronously. This is without much loss of generality, because the users must at least be moderately synchronized in reality, too, and synchronization is not really a task of the signature scheme. Moreover, for simplicity we permit only one transaction at each access point at a given time (whereas transactions at different access points may overlap), so that the entities of the signature scheme need not bother with scheduling.

4.5.3 Example of the Formalization

The requirement of the recipient on disputes is now presented formally, but, for readability, not from scratch in all details. The rest of formalization should then be fairly clear, except for availability of service. More details can be found in [P94].

The exact type of temporal logic we use is defined in [MP91]. We give an informal explanation of each formula and write the operators with their names, such as SINCE, instead of their symbols, so that readers unfamiliar with temporal logic can guess what is meant. We define one abbreviation:

$$f \text{ MODIFIED\_SINCE } g \;:=\; g \vee \text{PREVIOUS } (f \text{ SINCE } (f \wedge g)),$$

i.e., f is no longer required to hold in the current round, but must already have started in the round where g held.

The set of events at the user interface in each round is denoted by a variable *events*. Thus the requirements deal with the sequence of the values of the two variables *events* and *connections*.

Basic predicates about transactions. An input is valid if no other transaction is currently being executed at the same access point, i.e., no other input has been made there since the last output or since the system start. This form of "since" is precisely what the operator BACK_TO expresses. Hence we define

$$\begin{aligned} \mathit{valid\_input}(id, value) := \;& (id, in, value) \in \mathit{events} \\ & \wedge \text{WEAK\_PREVIOUS } \big((\neg \mathit{input\_at}(id)) \text{ BACK\_TO } \mathit{output\_at}(id)\big), \end{aligned}$$

where

$$\begin{aligned} \mathit{input\_at}(id) &:= \exists value'\!: (id, in, value') \in \mathit{events}; \\ \mathit{output\_at}(id) &:= \exists value'\!: (id, out, value') \in \mathit{events}. \end{aligned}$$

The operator WEAK_PREVIOUS, like PREVIOUS, denotes that something happened in the previous round, but it evaluates to TRUE in the very first round.

Similarly, if an output is considered, the value *value* of the previous valid input at the same access point, i.e., the user input on which the system is currently reacting, is characterized by

$$\begin{aligned} &\mathit{previous\_valid\_input}(id, value) := \\ &\qquad \neg \mathit{output\_at}(id) \text{ MODIFIED\_SINCE } \mathit{valid\_input}(id, value). \end{aligned}$$

Finally, we define that no output occurs at a set of access points, usually an interest group:

$$\mathit{no\_output\_at}(\mathit{interest\_group}) \;:=\; \forall id \in \mathit{interest\_group}\!: \neg \mathit{output\_at}(id).$$

Simple predicates about accepting and disputes. We first define that a recipient id_R accepts a message m as authenticated by a signer id_S:

$$\begin{aligned} \mathit{recipient\_accepts}(id_R, m, id_S) := \;& (id_R, out, \text{TRUE}) \in \mathit{events} \\ & \wedge \mathit{previous\_valid\_input}(id_R, (\text{'}test\text{'}, m, id_S)). \end{aligned}$$

The next predicate denotes that a recipient id_R and a third party id_T start a dispute about the message m consistently (and with a certain parameter ids_R):

$$\begin{aligned} \mathit{shows\_to}(id_R, id_T, m, id_S, ids_R) := \;& \mathit{valid\_input}(id_R, (\text{'}show\text{'}, m, id_S)) \\ & \wedge \mathit{valid\_input}(id_T, (\text{'}decide\text{'}, m, id_S, ids_R)). \end{aligned}$$

The desired results of such a dispute, in the interest of the recipient id_R and the third party id_T, are modeled next; the word "weakly" stands for the possibility of the output *'not_cooperated'*.

$$\begin{aligned} \textit{recipient\_weakly\_wins}(id_R, id_T) := \; & \big((id_T, out, \text{TRUE}) \in events \\ & \;\vee (id_T, out, \textit{'not\_cooperated'}) \in events\big) \\ & \wedge \exists eot\colon (id_R, out, eot) \in events. \end{aligned}$$

Sketch of connections and initialization. For readability, similar definitions of the following predicates are omitted:

- $\textit{correct\_init\_use}(\{id_R, id_T\}, id_S, ids_{out}, N)$, which means that the recipient and the third party have already carried out initialization for id_S consistently and successfully with the message bound N and the output ids_{out}, and
- $\textit{second\_init}(\{id_R, id_T\}, id_S))$, which means that the recipient or the third party try a second initialization for the signer id_S.
- $\textit{properly\_connected}(\textit{'dispute'}, \{id_R, id_T\})$, which means that the recipient and the third party are properly connected for a dispute in the current round.

Assembling the requirement. Our last predicate means that *if* a dispute is started correctly in the current round and correct connections are provided, it yields the correct result. The timing is as follows: There should be no other output until the correct result, except that it is no longer specified what the system does if the connections were wrong in the previous round. There is an exception to the exception, however, because the result might occur in the very first round.

$$\begin{aligned} & \textit{recipient\_can\_weakly\_convince}(id_R, id_T, m, id_S, ids_R) := \\ & \qquad \textit{shows\_to}(id_R, id_T, m, id_S, ids_R) \\ & \qquad \to \big(\textit{recipient\_weakly\_wins}(id_R, id_T) \\ & \qquad\quad \vee\; \textit{no\_output\_at}(\{id_R, id_T\}) \\ & \qquad\qquad \wedge \text{WEAK\_NEXT} \\ & \qquad\qquad\quad \big(\textit{no\_output\_at}(\{id_R, id_T\}) \\ & \qquad\qquad\quad\; \text{UNLESS } (\textit{recipient\_weakly\_wins}(id_R, id_T) \\ & \qquad\qquad\qquad\qquad \vee \text{ PREVIOUS } \neg\textit{properly\_connected}(\textit{'dispute'}, \{id_R, id_T\}))\big)\big). \end{aligned}$$

Finally, the requirement can be assembled. The recipient and the third party, i.e., the interest group, are formal parameters. The signer and the message are quantified universally. The requirement says that whenever the recipient has accepted a message, he can from then on convince the third party in the sense defined above, provided that both transactions occur in the state *correct_init_use*. Recall that the predicate *suitable* defined what parameters ids_R can be used.

$$\begin{aligned} & Req_{rec}(id_R, id_T) := \\ & \quad \forall id_S \in Id_S\; \forall m \in M\; \forall ids_{out} \in \mathcal{P}(Id)\; \forall N \in Bounds\colon \\ & \quad \text{HENCEFORTH} \;\; \big((\textit{recipient\_accepts}(id_R, m, id_S) \\ & \qquad\qquad \wedge\; \textit{correct\_init\_use}(\{id_R, id_T\}, id_S, ids_{out}, N)) \\ & \qquad\quad \to \text{NEXT } \forall ids_R\colon \\ & \qquad\qquad\qquad \big((\textit{suitable}(Id_R, ids_{out}, ids_R) \wedge id_R \in ids_R) \to \\ & \qquad\qquad\qquad (\textit{recipient\_can\_weakly\_convince}(id_R, id_T, m, id_S, ids_R) \\ & \qquad\qquad\qquad \text{UNLESS } \textit{second\_init}(\{id_R, id_T\}, id_S)))\big)\big). \end{aligned}$$

If one considers finite time, an operator WEAK_NEXT that evaluates to TRUE in the last round is needed instead of NEXT. The second initialization with the same signer has to be excluded because inside the system, it would lead to keys that do not fit the old signature. Of course, one can still perform a new initialization for the same *user*, e.g., by adding a time stamp to the identity.

4.6 Additional Service Properties

Additional service properties, which can be used to classify signature schemes, can be sorted as follows.

Stronger requirements, i.e., concretizations where the minimal specification is ambiguous. The following types exist in actual schemes:

- *Strong requirement of the recipient on disputes.* The recipient wishes that any message he has accepted should also be accepted by a third party in a subsequent dispute. Recall that a weaker minimal requirement was made because of undeniable signature schemes. Most signature schemes, however, fulfil the recipient's original wish. This is formalized by replacing the predicate *recipient_weakly_wins* by a predicate *recipient_wins* that does not contain the term "$\vee$ (id_T, *out*, *'not_cooperated'*) $\in$ *events*".
    - *Strong requirement of the signer on disputes.* The minimal requirement of the signer only guarantees her anything if she disavows the message. The corresponding strong requirement omits this precondition. In this case, one need not find the signer to carry out a fair dispute.
    - *Medium requirement of the signer on disputes.* An intermediate form is that if the signer does not take part in a dispute, *acc* may be FALSE or *'not_cooperated'*, but not TRUE, i.e., the message is not simply accepted. This is what undeniable signature schemes offer.
    - *User-friendliness*, i.e., requirements on the reaction of the system on inputs that can be regarded as user errors.

One can also define *directedness of authentication* to mean that the authentication is only valid for the intended recipients.

Multiple specifications. As specifications can be fulfilled with different degrees of security, it can make sense to have more than one specification. In particular, one can make strong requirements that will be guaranteed with a low degree of security, and weaker so called fall-back requirements that can be guaranteed with a higher degree of security, i.e., under more circumstances. In particular, the strong requirements might only be fulfilled under computational assumptions, but the fall-back requirement information-theoretically.

The *fail-stop* property is the only current example of multiple specifications. Informally, one first requires that no forgery ever occurs, but then, if a forgery occurs nevertheless, it should at least be provable. In the specification, this means that fall-back versions of both requirements on disputes are made, where both the signer and the recipient tolerate that the third party's output may be *'broken'*. An additional requirement for the low degree of security is that the output *'broken'* never occurs, i.e., the system is not falsely stopped. It is made in the interest of only the third party in that dispute, i.e., even if the signer and the recipient collude.

Special specification parameters are also a type of concretization where the minimal specification is ambiguous, see Section 4.4. Examples are:

- Special message spaces.
- Special dependencies on recipients. For instance, if the domain of ids_R only consists of sets with one element, signatures are specific to certain recipients. In contrast, in most signature schemes, ids_R always has to be ε, i.e., the signer cannot restrict the recipients.

Additional transactions, i.e., extensions of the domains of interface events, and corresponding requirements. Some existing ones are:

- A transaction *transfer* for a recipient to pass an authenticated message to a second recipient so that the second recipient is also sure that he can win disputes. This can work over arbitrary chains of recipients or only chains of fixed length. Most digital signature schemes permit arbitrary transfers, but undeniable ones none at all, and different versions of unconditionally secure ones none or only over chains of fixed length.
- If there is a fail-stop property, one usually also has a transaction to transfer the knowledge that the system has been broken.
- There can be more versions of initialization, e.g., one for new recipients to enter the system later. This is easy with ordinary digital signature schemes, but not with unconditionally secure schemes, where key generation is interactive.

One can also consider local transactions, such as queries about the status of an access points, and some distributed forms of the three minimal transactions, i.e., versions with more participants.

Privacy requirements. The invisibility of undeniable signatures is the only privacy requirement that has so far been considered with signature schemes. (Recall that blind signatures need a separate specification.) Note that invisibility is now an addition to the general definition, but cannot be formalized in the same style as the other additional properties.

5 Structure

As the service of digital signature schemes has been defined at an interface, one can consider implementations with arbitrary structure, e.g., even centralized ones. However, only decentralized ones are defined to *be* signature schemes: Two people meeting in the desert (with their hand-held computers) should be able to exchange authenticated messages. More precisely,

- there must be one entity per access point, as in Figure 2,
- and only the entities under the access points concerned take part in a transaction.

We call this *locality*.

5.1 General Structure

A signature **scheme** consists of the programs for such entities and the specification parameters, see Section 4.4, i.e., *Scheme* = (*spec_pars*, *signer_program*, *recipient_program*, *third_party_program*).

The "programs" are probabilistic polynomial-time interactive algorithms, which must distinguish in- and outputs at the user interface and on an arbitrary number of connections. The usual formalization in cryptology is interactive Turing machines. Note that these programs contain the real cryptologic algorithms, in contrast to system models used with logics of authentication.

A **system** is an instantiation of a scheme with certain system parameters *sys_pars*. These are the sets of identities (Id_S, Id_R, Id_T) and additional *security parameters*. Given a scheme and the system parameters, the actual system is defined as follows: There is one entity per identity, each running the appropriate program. Each entity has all the system parameters and its own identity as an initial state. The varying connections are handled by one additional entity, called switch, that models the network in a simple way: Its input is the variable *connections*, see Section 4.5.2, and it switches these connections according to the value of the specification parameter *channel_types*.

With any precise notion of an interactive algorithm, it is clear how global runs of such a system in interaction with users are defined.

Note that all the structural differences mentioned in Section 1.3 are nicely hidden inside this definition of a scheme: Within a transaction, the entities can carry out an arbitrary number of algorithms or interactive subprotocols, and they can use an arbitrary number of local variables from their memory.

5.2 Simpler Structures

Special structural properties are restrictions on the general structure. The most important one is non-interactive authentication, in the authentication transaction, only one message is sent from the signer's entity to the recipient's entity. These are the signature schemes where *signatures* in the normal sense exist. Generally, one can restrict any type of complexity, e.g., the number of rounds, the number of bits exchanged, or the storage used, and for each transaction type separately. Moreover, one can redefine notions such as public keys under certain structural restrictions.

6 Security

Security of a scheme with respect to a service specification means that the systems derived from the scheme only produce sequences of interface events permitted by the service specification, but, in contrast to usual program validity of temporal logic, even in the presence of certain attackers and possibly with small error probabilities.

6.1 General Security Definitions

6.1.1 Access of Attackers to Parts of the System

When a requirement in the interest of certain parties, such as one recipient and one third party, is considered, only the entities of these parties are assumed to be correct, i.e., to execute the programs from the scheme. Additionally, the channels between them are secure according to the their type — recall that only connections that are explicitly required to be correct are modeled in the input *connections* and its instantiation via *channel_types*.

With the minimal requirements on signature schemes, all attackers are assumed to collude, i.e., all incorrect entities are replaced by one large attacker entity.

6.1.2 Influence of Attackers on Honest Users

Sequences of interface events are produced by interaction of the system with arbitrary users. Even honest users who are interested parties may be influenced by the attacker, e.g., in the choice of messages they authenticate. An important case is the adaptive chosen-message attack on a signer discussed in Section 1.1. In more general signature schemes, active attacks on recipients and third parties are also possible and dangerous, because all entities may have secret information — this was originally overlooked with unconditionally secure schemes. The most general active attack on any reactive system is shown in Figures 3 and 4. In the more intuitive version in Figure 3, there is a universal quantifier over honest users who are influenced by the attacker. For requirements that only specify correct outputs, one can easily show that the simpler version in Figure 4, where the attacker has direct access to the access points above the correct entities, is equivalent.

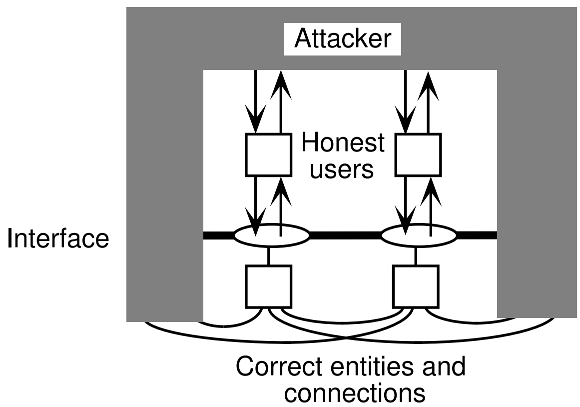

Figure 3 General active attack on two correct entities in a reactive system, version with indirect access

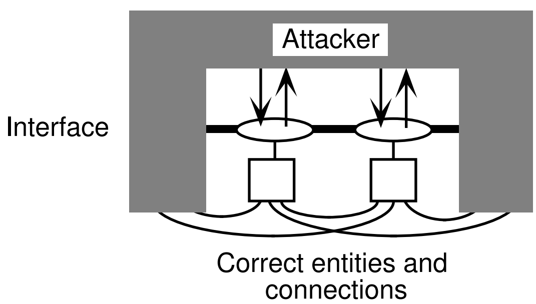

Figure 4 General active attack on two correct entities in a reactive system, version with direct access

6.1.3 Computational Aspects

For information-theoretical security, we simply allow the attacker with the structure of Figure 3 or 4 to choose an arbitrary probabilistic strategy. For computational security, the attacker is restricted to probabilistic polynomial-time computations.

6.1.4 Fulfilling and Error Probabilities

The fact that a scheme fulfils a logic requirement with information-theoretic security without error probability is now easily defined: The corresponding systems in interaction with an arbitrary attacker and arbitrary users in the structure of Figure 3 or 4 must only produce permitted sequences of interface events.

Information-theoretic security with a specific small probability of error, say 2^{-100}, is similarly straightforward, i.e., with this probability, the resulting sequence of interface events may be wrong. Note that the probabilities are taken over all the random choices of the correct entities and the attacker strategy. Usually, one does not fix one such error probability, but requires exponential decrease in a security parameter. Recall that security parameters are system parameters and present in the initial state of each entity, see Section 5.1.

If the class of attackers is computationally restricted, one can usually only require superpolynomial decrease of the error probability in a security parameter k, i.e., on requires that for all $c > 0$ and all attackers from the given class, there exists $k_0 \in \mathbb{N}$ such that for all $k \geq k_0$, the error probability is less than k^{-c}. There are only some difficulties with general dependencies of system parameters if more than one of them tends to infinity.

If the notion of “small error probability” is invariant against addition, then with two requirements, their conjunction is also fulfilled, so that the usual logic calculus is sound in the resulting cryptologic semantics.

6.2 Combinations of Degrees of Security

Each requirement on a digital signature scheme can be fulfilled with a different degree of security. An important classification criterion for signature schemes is the degrees of security of the two requirements on disputes. The following combinations exist, where dual security is a new term.

- **Ordinary security** means that the requirement of the signer on disputes holds computationally and that of the recipient information-theoretically. For instance, ordinary digital signature schemes and most undeniable signature schemes offer this type of security. (Recall that the test function of ordinary digital signature schemes is deterministic, hence a recipient who has accepted a signature is absolutely certain that any honest third party will also accept it.)
- **Dual security** is the other way round: The requirement of the signer on disputes holds information-theoretically and that of the recipient computationally. This kind of security is offered by the undeniable signature schemes in [CHP92]. Note that it is now very easy to describe this degree of security and its difference to fail-stop security, which it wasn’t before.

- **Unconditional security** means that both requirements on disputes hold information-theoretically.
- **Fail-stop security** means that both original requirements on disputes hold computationally and the fall-back requirements from Section 4.6 are fulfilled information-theoretically. It is now easy to see that fail-stop security is stronger than both ordinary and dual security: By identifying the output *'broken'* with FALSE, i.e., signatures are simply rejected if the computational assumption has been broken, one obtains dual security, and by identifying it with TRUE, i.e., signatures are still accepted even if the assumption has been broken, one obtains ordinary security.

Note, however, that one also has to differentiate whether the minimal or the strong requirements of the signer and the recipient on disputes are considered. In particular, simple versions of fail-stop signature schemes only guarantee the weak requirement of the signer in both the original and the fall-back version, but one can extend them so that the strong requirement of the signer is at least guaranteed computationally.

Furthermore, one has to mention which of the information-theoretic parts have error probabilities (always exponentially small). It can be shown that error probabilities cannot be avoided with the requirement of the signer on disputes in dual and fail-stop security and with both requirements on disputes in unconditional security [P94].

7 Conclusion

It has been shown that a general definition of signature schemes, covering all new types of them, can be given by the following three parts:

- *Service:* requirements in temporal logic about the behavior of such schemes at a user interface.
- *Structure*: mainly three programs for entities of a distributed system, instead of the individual algorithms carried out in them.
- *Degree of security*: rather general definitions of what it means that a scheme fulfils a logic requirement in the presence of certain classes of attackers (both with access to some system parts and some influence on honest users) in the information-theoretic and computational sense.

Based on this definition, a systematic classification of additional properties of special types of signature schemes has been shown.

Parts of this approach have only been sketched, in particular the definitions of schemes and degrees of security, which are not special to signature schemes, and some theorems about general signature schemes. They can hopefully be presented in more detail in future work.

Acknowledgments

I am happy to thank *Johannes Buchmann* for first drawing my attention to the lack of a general definition of signature schemes, *Joachim Biskup*, *Gerrit Bleumer*, *Andreas Pfitzmann*, and

Michael Waidner for interesting and helpful discussions about this paper, and *Michaela Huhn* and *Peter Niebert* for helping my understanding of temporal logic.

References

[AT91] Martín Abadi, Mark R. Tuttle: A Semantics for a Logic of Authentication; 10th PoDC, acm press, New York 1991, 201-216.

[B91] Donald Beaver: Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority; J. Cryptology 4/2 (1991) 75-122.

[BAN89] Michael Burrows, Martín Abadi, Roger Needham: A Logic of Authentication; ACM Operating Systems Review 23/5 (1989) 1-13.

[BC93] Jurjen Bos, David Chaum: Provably Unforgeable Signatures; Crypto '92, LNCS 740, Springer Verlag, Berlin 1993, 1-14.

[BCDP91] Joan Boyar, David Chaum, Ivan Damgård, Torben Pedersen: Convertible Undeniable Signatures; Crypto '90, Springer-Verlag, Berlin 1991, 189-205.

[BM92] Mihir Bellare, Silvio Micali: How to Sign Given Any Trapdoor Permutation; Journal of the ACM 39/1 (1992) 214-233.

[BPW91] Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A remark on a signature scheme where forgery can be proved; Eurocrypt '90, Springer-Verlag, Berlin 1991, 441-445.

[C85] David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.

[C91] David Chaum: Zero-knowledge undeniable signatures; Eurocrypt '90, Springer-Verlag, Berlin 1991, 458-464.

[C95] David Chaum: Designated Confirmer Signatures; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 86-91.

[CA90] David Chaum, Hans van Antwerpen: Undeniable signatures; Crypto '89, Springer-Verlag, Heidelberg 1990, 212-216.

[CD95] Ronald Cramer, Ivan Damgård: Secure Signature Schemes based on Interactive Protocols; Crypto '95, Proceedings, LNCS 963, Springer-Verlag, Berlin 1995, 297-310.

[CH91] David Chaum, Eugène van Heijst: Group signatures; Eurocrypt '91, LNCS 547, Springer-Verlag, Berlin 1991, 257-265.

[CHP92] David Chaum, Eugène van Heijst, Birgit Pfitzmann: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer; Crypto '91, Springer-Verlag, Berlin 1992, 470-484.

[CR91] David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Crypto '90, Springer-Verlag, Berlin 1991, 206-214.

[D84] Dorothy E. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems; CACM 27/4 (1984) 388-392.

[DF92] Yvo Desmedt, Yair Frankel: Shared generation of authenticators and signatures; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 457-469.

[DH76] Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Trans. Inform. Theory 22/6 (1976) 644-654.

[DN94] Cynthia Dwork, Moni Naor: An efficient existentially unforgeable signature scheme and its applications; Crypto '94, LNCS 839, Springer-Verlag, Berlin 1994, 234-246.

[DPP94] Ivan B. Damgård, Torben P. Pedersen, Birgit Pfitzmann: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994, 250-265.

[DSS92] The Digital Signature Standard Proposed by NIST; CACM 35/7 (1992) 36-40.

[DY81] Danny Dolev, Andrew C. Yao: On the Security of Public Key Protocols; FOCS 1981, IEEE Computer Society, 1981, 350-357.

[FGY92] Matthew Franklin, Zvi Galil, Moti Yung: An Overview of Secure Distributed Computing; Columbia University, Dep. of Computer Science, TR CUCS-008-92, March 24, 1992.

[FP91] Dirk Fox, Birgit Pfitzmann: Effiziente Software-Implementierung des GMR-Signatursystems; Proc. Verläßliche Informationssysteme (VIS'91), Informatik-Fachberichte 271, Springer-Verlag, Heidelberg 1991, 329-345.

[G87] Oded Goldreich: Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 104-110.

[GM95] Rosario Gennaro, Silvio Micali: Verifiable Secret Sharing as Secure Computation; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 168-182.

[GMR88] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

[GMW87] Oded Goldreich, Silvio Micali, Avi Wigderson: How to play any mental game – or – a completeness theorem for protocols with honest majority; STOC 1987, ACM, New York 1987, 218-229.

[GQ91] Louis Claude Guillou, Jean-Jacques Quisquater: Precautions taken against various potential attacks in ISO/IEC DIS 9796; Eurocrypt '90, Springer-Verlag, Berlin 1991, 465-473.

[HP93] Eugène van Heyst, Torben P. Pedersen: How to make efficient fail-stop signatures; Eurocrypt '92, Springer-Verlag, Berlin 1993, 366-377.

[HPP93] Eugène van Heijst, Torben Pryds Pedersen, Birgit Pfitzmann: New Constructions of Fail-Stop Signatures and Lower Bounds; Crypto '92, LNCS 740, Springer Verlag, Berlin 1993, 15-30.

[ISO91] ISO (International Organization for Standardization): Digital signature scheme giving message recovery; International Standard ISO/IEC 9796, Preliminary edition 1991-04-22.

[L83] Peter F. Linington: Fundamentals of the Layer Service Definitions and Protocol Specifications; Proc. IEEE 71/12 (1983) 1341-1345.

[LABW92] Butler Lampson, Martín Abadi, Michael Burrows, Edward Wobber: Authentication in Distributed Systems: Theory and Practice; ACM Trans. Comput. Syst. 10/4 (1992) 265-310.

[M95] Catherine A. Meadows: Formal Verification of Cryptographic Protocols: A Survey; Asiacrypt '94, LNCS 917, Springer-Verlag, Berlin 1995, 133-150.

[MP91] Zohar Manna, Amir Pnueli: The Temporal Logic of Reactive and Concurrent Systems – Specification –; Springer-Verlag, New York 1991.

[MR91] Silvio Micali, Phillip Rogaway: Secure Computation (Ch. 1-3); Laboratory for Computer Science, MIT, Cambridge, MA 02139, USA; distributed at Crypto '91.

[NY89] Moni Naor, Moti Yung: Universal One-way Hash Functions and their Cryptographic Applications; 21st STOC, ACM, New York 1989, 33-43.

[P94] Birgit Pfitzmann: Fail-Stop Signature Schemes; Dissertation, Universität Hildesheim, 20.5.1994.

[Pe91] Torben Pryds Pedersen: Distributed Provers with Applications to Undeniable Signatures; Eurocrypt '91, LNCS 547, Springer-Verlag, Berlin 1991, 221-242.

[Pf91] Birgit Pfitzmann: Fail-stop Signatures; Principles and Applications; Compsec '91, Elsevier, Oxford 1991, 125-134.

[PP95] Torben Pryds Pedersen, Birgit Pfitzmann: Fail-Stop Signatures; accepted for SIAM Journal on Computing, revised version, 21.3.1995.

[PW90] Birgit Pfitzmann, Michael Waidner: Formal Aspects of Fail-stop Signatures; Interner Bericht 22/90, Fakultät für Informatik, Universität Karlsruhe, December 1990.

[PW92] Birgit Pfitzmann, Michael Waidner: Unconditional Byzantine Agreement for any Number of Faulty Processors; STACS 92, Springer-Verlag, Heidelberg 1992, 339-350.

[PW94] Birgit Pfitzmann, Michael Waidner: A General Framework for Formal Notions of "Secure" System; Hildesheimer Informatik-Berichte 11/94, ISSN 0941-3014, Institut für Informatik, Universität Hildesheim, April 1994.

[R90] John Rompel: One-Way Functions are Necessary and Sufficient for Secure Signatures; 22nd STOC, ACM, New York 1990, 387-394.

[RSA78] Ronald L. Rivest, Adi Shamir, Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; CACM 21/2 (1978) 120-126.

[S85] Adi Shamir: Identity-Based Cryptosystems and Signature Schemes; Crypto '84, LNCS 196, Springer-Verlag, Berlin 1985, 47-53.

[S91] Claus P. Schnorr: Efficient Signature Generation by Smart Cards; J. Cryptology 4/3 (1991) 161-174.

[SM93] Paul Syverson, Catherine Meadows: A Logical Language for Specifying Cryptographic Protocol Requirements; Proc. 1993 IEEE Symp. on Security and Privacy, Oakland, California, 165-177.

[Y82] Andrew C. Yao: Protocols for Secure Computations; 23rd FOCS, IEEE Computer Society, 1982, 160-164.