Tailoring Authentication Protocols to Match Underlying Mechanisms*

Liqun Chen, Dieter Gollmann and Christopher J. Mitchell

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
E-mail:{liqun,dieter,cjm}@dcs.rhbnc.ac.uk

Abstract. Authentication protocols are constructed using certain fundamental security mechanisms. This paper discusses how the properties of the underlying mechanisms affect the design of authentication protocols. We firstly illustrate factors affecting the selection of protocols generally. These factors include the properties of the environment for authentication protocols and the resources of the authenticating entities. We then consider a number of authentication protocols which are based on mechanisms satisfying different conditions than those required for the ISO/IEC 9798 protocols, in particular the use of non-random nonces and the provision of identity privacy for the communicating parties.

1 Introduction

This paper is concerned with the fundamental security mechanisms which are used to construct entity authentication protocols, and how the strength of these mechanisms affects protocol design. For the purposes of this paper, the mechanisms are divided into two categories, cryptographic mechanisms and time variant parameters (TVPs). The first category includes symmetric encryption, digital signatures, cryptographic check functions (CCFs) and zero knowledge techniques. The second includes clocks for timestamping, nonce generators and sequence numbers. An authentication protocol is typically constructed using at least one mechanism from both categories.

When a new protocol is needed in a particular environment, the designer must first discover what mechanisms are available. For implementation reasons there may be limits on available mechanisms, because every mechanism has particular properties corresponding to the particular environment in which the protocol works. In such a case, the protocol must meet the needs of the environment and be tailored to the 'strength' of the available mechanisms.

A variety of entity authentication protocols[2] have recently been standardised by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The four published parts of ISO/IEC

* This work has been jointly funded by the UK EPSRC under research grant GR/J17173 and the European Commission under ACTS project AC095 (ASPeCT).

[2] They are referred to as entity authentication mechanisms in ISO/IEC documents.

9798 cover the general model (9798-1, [1]), authentication protocols based on symmetric encipherment (9798-2, [3]), authentication protocols based on public key algorithms (9798-3, [2]), and authentication protocols based on CCFs (9798-4, [4]). A fifth part of ISO/IEC 9798, covering authentication protocols using zero knowledge techniques, is currently at CD stage [5]. The ISO/IEC 9798 protocols use TVPs, e.g. timestamps, nonces and sequence numbers, to prevent valid authentication information from being accepted at a later time.

The protocols proposed in ISO/IEC 9798 have all been designed to use mechanisms meeting rather stringent requirements, which may be difficult to meet in some practical environments. Hence in this paper we look at alternatives to the ISO/IEC 9798 protocols which are still sound even when the mechanisms do not meet such 'strong' conditions.

The discussion starts in Section 2 with an illustration of factors which affect protocol selection, including the properties of the environment for an authentication protocol, and the resources of the authenticating entities. We then look at the ISO/IEC 9798 protocols in the context of requirements on the mechanisms. In Sections 3, 4, 5, 6, 7 and 8, we consider possible alternatives to the ISO/IEC 9798 schemes, which do not put such strict conditions on the mechanisms. In the final section, we give a summary of the contributions of this paper.

2 Factors affecting protocol selection

We now discuss two important factors which affect protocol design, namely the properties of the protocol environment and the resources of the authenticating entities. As we show in subsequent sections, if the environment and entity resources for authentication satisfy less stringent conditions than those required for the ISO/IEC 9798 protocols, then the ISO/IEC 9798 protocols cannot be used, and a different protocol, tailored to match the properties of the underlying mechanisms, needs to be designed.

2.1 Properties of the environment for protocols

Before considering what underlying mechanisms are available and selecting a protocol which uses them, the designer must know the environment in which the protocol will work. A particular environment may impose stringent requirements on the mechanisms and the protocol itself. For the purposes of this paper we consider the following aspects of the environment of an authentication protocol.

Communications channel. Communications channels are used to transmit messages during the process of authentication. The major property of interest here is whether a channel is broadcast or point-to-point.

◇ *Broadcast channel*, where there exist messages from a variety of senders and/or for a variety of receivers. Typically, to make a broadcast channel operate correctly, the sender's and/or receiver's names have to be sent across the channel.

◇ *Point-to-point channel*, where the channel is reserved for a particular sender and receiver who both know the initiator and terminator of each message, so that their names do not need to be sent across the channel.

Other properties of the communications channel which affect the design of authentication protocol include whether or not interceptors can modify messages and/or introduce totally spurious messages. However we do not address these issues in detail here.

Entity identifier. The major property of interest here is which entities are authorised to know a particular entity identifier during the process of authentication. There are two main cases of interest, namely that the identifier of an entity is allowed to be transferred in clear text (e.g., ISO/IEC 9798 parts 2, 3 and 4) or that it is only allowed to be known to particular entities (e.g., [6, 13, 17]).

Trust relationship. The trust relationship describes whether one entity believes that the other entities will follow the protocol specifications correctly (e.g. [14]). The trust relationship of particular interest here is between authentication servers and clients. For instance, a client may not trust an individual server [9].

2.2 Resources of authenticating entities

We consider the following aspects of the resources of the authenticating entities.

Knowledge. An entity might have different kinds of knowledge at the start of authentication, namely shared secret information with another entity, private information (e.g. the private part of the entity's own asymmetric key pair), reliable public information (e.g. the public part of another entity's asymmetric key pair), or no knowledge of another entity.

Computational ability. The entities may or may not have the computational ability to perform certain operations, namely computation of complex cryptographic algorithms (e.g. a digital signature algorithm), random and secure generation of a key or an offset for a key, and generation of predictable or unpredictable nonces.

Time synchronisation. The entities may or may not have access to securely synchronised clocks or synchronised sequence numbers.

3 Absence of trust in individual third parties

We consider a situation where two entities, who share no secret based on symmetric cryptography or hold no public information based on asymmetric cryptography, want to complete unilateral or mutual authentication. Typically they will have to get assistance from a third party, referred to as an authentication server. However, in some environments, these clients have no reason to trust an individual server, and in such a case the ISO/IEC 9798 protocols cannot be used directly. In order to design a protocol which does not require trust in individual servers, a range of possible approaches can be followed.

Firstly, a client can choose which from a set of servers to trust, typically by applying a security policy or considering their history of performance and reliability. Yahalom et al. [21] proposed a protocol which allows a client or his agent to make such a choice. A limitation of this scheme is that a client may sometimes find it difficult to distinguish between trustworthy and untrustworthy servers.

Secondly, a set of moderately trusted servers who are trusted by users collectively, but not individually, can be used. Gong [11] proposed a protocol with multiple servers such that a minority of malicious and colluding servers cannot compromise security or disrupt service. In that protocol two clients participate in choosing a session key, and each relevant server is responsible for converting and distributing a part of the session key. Two variations on this approach have been described in [9]. In both variants the servers each generate a part of a session key, which can be successfully established between a pair of entities as long as more than half the servers are trustworthy. Both schemes from [9] have the advantage of requiring considerably fewer messages than Gong's protocol.

A third approach, based on asymmetric cryptography, is to separate transfer of authentication information from issue of authentication information, i.e. to let the issuer be off-line. One instance of this approach is where a master server (sometimes called the certification authority) issues a certificate which is then held by another on-line secondary server. The certificate is valid for a period of time, during which there is no need to further contact the master server, since the certificate is available from the secondary server. The client does not need to trust the secondary server, but does need to trust the off-line master server.

4 Entity identity privacy

During authentication, the identities of the entities concerned may need to be sent across the communications channel, either embedded within or alongside the protocol messages. There are two main reasons why this may be necessary.

- Depending on the nature of the communications channel, all messages may need to have one or more addresses attached. More specifically, if a broadcast channel is being used, then, in order for recipients to detect the messages intended for them, a recipient address must be attached. In addition, many authentication protocols require the recipient of a protocol message to know the identity of the entity which (claims to have) originated it, so that it can be processed correctly (e.g. deciphered using the appropriate key). If this information is not available, as would typically be the case in a broadcast environment, then the originator address also needs to be attached.
- Certain authentication protocols, including some of those in ISO/IEC 9798, require the recipient's name to be included within the protected part of some of the messages, in order to protect the protocol against certain types of attack.

However, depending on the nature of the communications channel, communicating entities may require a level of anonymity, which would prevent their name and/or address being sent across the channel (except in enciphered form). For example, in a mobile telecommunications environment, it may be important for users that their identifiers are not sent in clear across the radio path, since that would reveal their location to an interceptor of the radio communications.

Of the two reasons listed for sending names across the channel, the second is usually simpler to deal with, since alternative protocols can typically be devised which do not require the inclusion of names in protocol messages; for example, as described in ISO/IEC 9798–4, 'unidirectional keys' can be employed. We therefore focus our attention on the anonymity problems arising from the use of authentication protocols in broadcast environments. It is important to note that these anonymity problems are not dealt with in any published part of ISO/IEC 9798.

There are two main approaches to providing entity anonymity in broadcast channels, i.e., the use of temporary identities which change at regular intervals, and the use of asymmetric (public key) encipherment techniques. We now discuss three examples which make use of these approaches. In each case we consider a 'many-to-one' broadcast scenario, where many mobile users communicate with a single 'base' entity. In this case anonymity is typically only required for the mobile users who can only receive from the base, and hence there may be no need for the base address to be sent across the channel.

First observe that in GSM (Global System for Mobile Telecommunication) [13], temporary identities (TMSIs) are transmitted over the air interface instead of permanent identities (IMSIs). TMSIs are changed on each location update and on certain other occasions as defined by the network. A mobile user identifies himself by sending the old TMSI during each location update process, which has to be sent before authentication takes place and must therefore be sent unencrypted. However, the new TMSI is returned after authentication has been completed and a new session key has been generated so that it can be, and is, transmitted encrypted. In certain exceptional cases, including initial location registration, the user has to identify itself using its IMSI. In these cases an intruder may be able to obtain the IMSI from the GSM air interface. Thus the GSM system only provides a limited level of anonymity for mobile users.

Second consider a mutual authentication protocol, also outlined in [17], which has been designed for possible adoption by two third generation mobile systems, namely UMTS (Universal Mobile Telecommunications System) and FPLMTS (Future Public Land Mobile Telecommunication Systems). Like the GSM solution, this scheme also uses temporary identities to provide identity and location privacy. However, in this protocol, temporary identities are used at every authentication exchange including the case of a new registration, so that permanent identities are never transmitted in clear text.

The principals involved in this protocol are one of number of users, A, a single 'base' entity B and an authentication server S. The protocol makes use of two types of temporary identities: S-identities shared by A and S, including a current I_A and a new I'_A, and B-identities shared by A and B, also including a current J_A and a new J'_A. There are two versions of the protocol, depending on whether or not A and B already share a valid temporary B-identity J_A and secret key K_{AB}. The protocol makes use of five cryptographic check functions $F1$ - $F5$, each of which takes a key and a data string as input. Note also that $\oplus$ denotes bit-wise exclusive-or.

Version 1: A and B share K_{AB} and J_A. Then the message exchanges are as follows (where $M_i : A \rightarrow B : x$ means that the ith exchanged message is sent from A to B and contains data x).

$$\begin{aligned}
&M_1 : A \rightarrow B : J_A, R_A \\
&M_2 : B \rightarrow A : R_B, F4_{K_{AB}}(R_A) \oplus J'_A, F3_{K_{AB}}(R_B, R_A, J'_A) \\
&M_3 : A \rightarrow B : F3_{K_{AB}}(R_A, R_B)
\end{aligned}$$

Version 2: A and B share no secret, A and S share K_{AS} and I_A, and B and S have a secure channel which is available for exchanging messages 2 and 3.

$$\begin{aligned}
&M_1 : A \rightarrow B : I_A, R_A \\
&M_2 : B \rightarrow S : I_A, R_A \\
&M_3 : S \rightarrow B : F1_{K_{AS}}(R_A) \oplus I'_A, O, K_{AB}, F3_{K_{AS}}(I'_A, R_A, O) \\
&M_4 : B \rightarrow A : F1_{K_{AS}}(R_A) \oplus I'_A, O, F3_{K_{AS}}(I'_A, R_A, O), R_B, \\
&\qquad\qquad\quad F4_{K_{AB}}(R_B) \oplus J'_A, F3_{K_{AB}}(R_B, R_A, J'_A) \\
&M_5 : A \rightarrow B : F3_{K_{AB}}(R_A, R_B)
\end{aligned}$$

where O is a key offset so that $K_{AB} = F2_{K_{AS}}(O, B)$. The resulting session key is $K'_{AB} = F5_{K_{AB}}(R_A, R_B, J'_A)$.

Our third example is based on the use of asymmetric encipherment. In this case, no temporary sender identity is required in the first message because the real sender identity can be encrypted using the public encipherment transformation of the receiver. In order to keep the receiver identity confidential, a temporary receiver identity is needed. In this example, A is one of a number of users of a single broadcast channel which is used for communicating with a single 'base' entity B, E_X denotes public key encipherment using the public key of entity X and H denotes a hash-function. The messages are:

$$\begin{aligned}
&M_1 : A \rightarrow B : E_B(A, R_A, R'_A) \\
&M_2 : B \rightarrow A : R'_A, E_A(R_B, R_A, B) \\
&M_3 : A \rightarrow B : H(R_A, R_B, A)
\end{aligned}$$

where the nonce R'_A is a temporary identifier for A, which should be unpredictable for any third parties. In this protocol, the nonces R_A and R_B are used to meet two requirements, namely to verify the freshness of messages by using a challenge-response scheme and, possibly, to establish a session key shared between A and B. These nonces have to be unpredictable as well. Note also that A and B must have reliable copies of each other's public keys before starting the protocol.

5 Avoiding abuse of digital signatures

During authentication based on digital signatures with 'unpredictable' nonces, entity A typically challenges entity B by sending a nonce R_A. B then sends A a signature-based message containing this nonce in reply to the challenge. By choosing the nonce appropriately A may be able to use B's signature for malicious purposes.

Here, as throughout this paper, a digital signature function is defined to include use of either a hash-function or a redundancy function to prevent an impersonator claiming that a randomly chosen value is actually a signature.

The authentication protocols of ISO/IEC 9798-3 [2] discuss means of dealing with this problem, and to help avoid the worst consequences a nonce chosen by the signer can also be included in the relevant signature. However, the same problem may still exist. We now consider a protocol given in Clause 5.2.2 of ISO/IEC 9798-3 (see also [18]).

$$\begin{aligned}
&M_1 : B \rightarrow A : R_B, D_1\\
&M_2 : A \rightarrow B : CertA, R_A, R_B, B, D_3, S_A(R_A, R_B, B, D_2)\\
&M_3 : B \rightarrow A : CertB, R_B, R_A, A, D_5, S_B(R_B, R_A, A, D_4)
\end{aligned}$$

Note that D_1 - D_5 are (application dependent) data strings, S_X denotes the signature function of entity X, R_A and R_B are nonces, and $CertX$ denotes the certificate of entity X. R_A is present in the signed part of M_2 to prevent B from obtaining the signature of A on data chosen by B. Similarly R_B is present in the signed part of M_3. However, this approach cannot completely avoid the abuse of signatures for the following two reasons.

1. Although both nonces are included in both signatures, B selects its nonce before A. This means that A is in a more favourable position than B to misuse the other party's signature. To prevent this, B can generate an extra nonce and add it into the signed message, e.g. in M_3 a nonce R'_B can be included in D_4 and D_5:

$$M_3 : B \rightarrow A : CertB, R_B, R_A, A, R'_B, D'_5, S_B(R_B, R_A, A, R'_B, D'_4).$$

2. It is possible for users of the signatures to 'bypass' some nonces involved in the protocol if other signatures in different contexts use nonces in the same way. For example, a different protocol might make use of a message $S_A(R, X, B, D_2)$ or $S_B(R, Y, A, D_4)$, where R is a random number and X or Y has a particular meaning. The signatures of the previous protocol could then potentially be successfully abused in this protocol.

The above discussion implies that changing protocol construction only makes abuse of digital signatures more difficult, and cannot protect against such attacks completely, because the protocol itself cannot detect the misuse of digital signatures involved. However, there exist means of avoiding these problems, such as the following.

- 'Key separation' is a well known and widely used technique, i.e. using different keys for different applications (see, e.g. [18]).
- Another approach is using sequence numbers rather than unpredictable nonces to control the uniqueness of authentication exchanges. Because the values of the sequence numbers are agreed by both the claimant and verifier, neither party to a protocol can be persuaded to sign information which has some 'hidden meaning'.
- Last, but not least, observe that zero knowledge protocols are specifically designed to prevent this type of attack because they do not generate digital signatures. However, as Bengio et al. pointed out, these protocols are still open to interactive abuse through middleperson attacks [8]. It is then left to the implementation to ensure that those attacks are not feasible in practice.

6 Predictable and unpredictable nonces

The nonce-based protocols specified in Parts 2, 3 and 4 of ISO/IEC 9798 all require the nonces used to be unpredictable, i.e. the nonces must be generated in such a way that intercepting third parties cannot guess future nonce values. However, in some circumstances it may be necessary to use nonces which are predictable, i.e. generated using a deterministic process known to the interceptor. For example, it may be difficult for an entity to generate random or unpredictable pseudo-random numbers, particularly if the entity is implemented in a portable device, such as a smart card, and use of a simple counter for nonce generation may be the only practical possibility.

We now consider how secure nonce-based authentication protocols can be devised even if the nonces are predictable (as long as they are still 'one time'). This can be achieved by cryptographically protecting all the messages in a protocol including the nonces.

Before proceeding we briefly distinguish between predictable nonces and sequence numbers, both of which are used to control the uniqueness of messages. They are both used only once within a valid period of time and there is the possibility that they can be predicted in advance by a third party. However, a predictable nonce is used as a challenge, so that the responder does not need to know it before receiving it and to record it after using it. On the other hand, a sequence number is agreed by the claimant and verifier beforehand according to some policy, and will be rejected if it is not in accordance with the agreed policy. Furthermore, use of sequence numbers may require additional 'book keeping' for both the claimant and verifier. Typically every entity will need to store a 'send' sequence number and a 'receive' sequence number for each other entity with which they communicate.

It has been observed in [12] that in a protocol using symmetric encryption with a nonce, if the nonce is unpredictable then either the challenge or the response can be transmitted unencrypted; however if the nonce is predictable then both the challenge and response have to be encrypted. Otherwise the protocol cannot protect against replay attacks.

We now illustrate that this logic also applies to protocols using digital signatures and CCFs. The following example shows how digital signatures can be used in conjunction with a predictable nonce to produce a secure authentication protocol. This is a modification of the protocol given in Clause 5.2.2 of [2] (the notation is as used previously).

$$\begin{aligned} M_1 &: B \rightarrow A : CertB, R_B, S_B(R_B) \\ M_2 &: A \rightarrow B : CertA, R_A, S_A(R_A, R_B, B) \\ M_3 &: B \rightarrow A : S_B(R_B, R_A, A) \end{aligned}$$

Another example, this time based on the use of a CCFF, is the following modification of the protocol given in Clause 5.2.2 of [4].

$$\begin{aligned} M_1 &: B \rightarrow A : R_B, F_K(R_B) \\ M_2 &: A \rightarrow B : R_A, F_K(R_A, R_B, B) \\ M_3 &: B \rightarrow A : F_K(R_B, R_A) \end{aligned}$$

The above analysis means that there is a good range of protocols available to support both unpredictable and predictable nonces. Note that when using a predictable nonce as a challenge, since a future challenge is predictable to the responder, the verifier of the challenge has to depend on the honesty and competence of the responder [12].

7 Disclosure of plaintext/ciphertext pairs

There are a number of different models for known plaintext attacks, chosen plaintext attacks and chosen ciphertext attacks on cipher systems (e.g. [7] and [10]). Although obtaining plaintext/ciphertext pairs far from guarantees that attacks will be successful, it is typically the necessary first step for an attacker. Whether or not the attacker has access to a plaintext/ciphertext pair during authentication depends on the nature of the communications channel, the authentication protocol, and the details of the cryptographic processing.

We use the term ‘plaintext/ciphertext pair’ rather loosely here, to cover matching pairs of input and output for a variety of cryptographic algorithms, including encipherment algorithms, digital signatures and cryptographic check functions. Whether or not disclosing a plaintext/ciphertext pair is a problem depends on the strength of the algorithm, and whether the same algorithm and key are used for applications other than the authentication exchange.

In some situations the disclosure of plaintext/ciphertext pairs is not a security problem and that is what ISO/IEC 9798 assumes. However we are concerned here with situations where disclosure of pairs may be a security threat, and we consider ways of avoiding the threat. The following unilateral authentication protocols are given in ISO/IEC 9798 parts 2, 3 and 4.

Example 1: Symmetric encryption with nonce [3]:

$$\begin{aligned} M_1 &: B \rightarrow A : R_B, D_1 \\ M_2 &: A \rightarrow B : D_3, E_{K_{AB}}(R_B, B, D_2) \end{aligned}$$

Example 2: Digital signature with timestamp [2]:

$$M_1 : A \rightarrow B : T_A, B, D_1, S_A(T_A, B, D_2)$$

Example 3: CCF with sequence number [4]:

$$M_1 : A \rightarrow B : N_A, B, D_1, F_{K_{AB}}(N_A, B, D_2)$$

The first protocol is based on symmetric encryption. It depends on the optional text field D_2 whether a plaintext/ciphertext pair can be obtained. If D_2 is predictable data (including a null string), the plaintext/ciphertext pair is exposed. If D_2 includes some unpredictable data not supplied in D_1, it depends only on the structure of the cryptographic operation whether the plaintext/ciphertext pair is exposed.

In both the second protocol based on digital signature and the third protocol based on CCF, D_2 cannot be unpredictable for B, otherwise B cannot verify the signature and the cryptographic check value respectively. Those two protocols expose plaintext/ciphertext pairs.

Observe different time variant parameters based authentication protocols without an unpredictable optional text field placed in the cryptographically protected part of the token. In timestamp or sequence number based protocols, it is rather difficult to avoid disclosure of plaintext/ciphertext pairs, since the intruder can choose when to start the protocol or what predictable number is used in the protocol. When using an unpredictable nonce, the nonce has to be cryptographically protected in order to prevent the protocol disclosing a plaintext/ciphertext pair. The following examples of unilateral authentication protocols, which do not disclose plaintext/ciphertext pairs, do not depend on the predictability of D_1, D_2 and D_3.

Example 4: Symmetric encryption with nonce:

$$\begin{aligned} &M_1 : B \rightarrow A : E_{K_{AB}}(R_B, D_1) \\ &M_2 : A \rightarrow B : D_3, E_{K_{AB}}(R_B, B, D_2) \end{aligned}$$

Example 5: Digital signature and asymmetric encipherment with nonce:

$$M_1 : A \rightarrow B : E_B(R_A, B, D_1, S_A(R_A, B, D_2))$$

Example 6: CCF with nonces:

$$M_1 : A \rightarrow B : R_A, B, D_1, F_{K_{AB}}(R_A, B, D_2) \oplus R'_A, F_{K_{AB}}(R'_A, D_3)$$

Note that the first nonce R_A in Example 6 can be replaced by a timestamp T_A or sequence number N_A.

8 Using poorly synchronised clocks

The authentication protocols with timestamps specified in ISO/IEC 9798 require the communicating parties to have synchronised clocks. There are several approaches to achieving secure clock synchronisation and re-synchronisation (e.g., [19, 20, 16]). However, in some environments, time stamp based protocols need to be used although the parties do not have exactly synchronised clocks. For example, in a mobile telecommunications system, a mobile user may find it difficult to keep a clock synchronised with the clock of its service provider. The user and network may still wish to use a timestamp-based authentication protocol rather than a nonce-based one to reduce the number of messages exchanged, and to allow the detection of forced delays.

Two points must be considered when using a timestamp-based protocol in such an environment. Firstly the size of the acceptance window to match poorly synchronised clocks must be selected. The size of this window can be either fixed or dynamically changed depending on the environment, in particular on the delays in the communications channels and the quality of the clocks in use. Secondly all messages within the current acceptance window must be logged, and second and subsequent occurrences of identical messages within that window must be rejected (see Annex B of [3] and [15]).

9 Summary

This paper discusses how the properties of underlying mechanisms affect the design of authentication protocols and how to tailor authentication protocols to match underlying mechanisms. A number of alternatives to the ISO/IEC 9798 protocols, which do not put such strict conditions on the underlying mechanisms, have been proposed and analysed. We now summarise them as follows.

- ◇ If authentication servers are not trusted by clients individually, three possible approaches are: (1) allowing clients to choose trustworthy servers, (2) using a set of moderately trusted servers instead of a single one, (3) using off-line master servers.
- ◇ In order to preserve entity identity privacy, two possible methods are: (1) based on asymmetric cryptography the entity identity can be hidden by using the public part of the receiver's asymmetric key pair, (2) one or more temporary entity identities instead of the real entity identity can be transmitted unencrypted.
- ◇ In order to avoid abuse of digital signatures, four possible approaches are: (1) let the signer generate an unpredictable nonce and insert the nonce into the signed message, (2) use different keys for different applications, (3) use sequence numbers rather than unpredictable nonces to control the uniqueness of authentication exchanges, (4) use zero knowledge techniques.
- ◇ When using a predictable nonce as a challenge, all messages, both the challenge and response, have to be protected cryptographically. Some possible examples are: (1) in a symmetric encryption based protocol, the challenge

and response are respectively a nonce and a function of the nonce encrypted by the shared key, (2) in a digital signature based protocol, the private parts of the challenger's and responder's asymmetric key pairs are used in the challenge and response respectively, (3) in a CCF based protocol, the challenge is a nonce concatenated with a CCF of the nonce, and the response is a CCF of a function of the nonce.

◇ In unpredictable nonce based protocols without unpredictable optional text fields it is possible to avoid disclosing plaintext/ciphertext pairs by using cryptographic protection for every message. In timestamp or sequence number based protocols without unpredictable optional text fields, it appears to be rather difficult to avoid giving plaintext/ciphertext pairs.

◇ When using poorly synchronised clocks in authentication protocols, one approach is to take the following steps: (1) select the size of the acceptance window to match the poorly synchronised clocks, (2) log all messages and reject the second and subsequent occurrences of identical messages within the acceptance window.

10 Acknowledgements

The authors would like to thank Volker Kessler and Günther Horn for a useful suggestion which improved the authentication protocol using asymmetric encipherment in Section 4, and thank anonymous referees and Anish Mathuria for careful comments on an earlier version of this paper.

References

1. ISO/IEC 9798-1: 1991. Information technology — Security techniques — Entity authentication mechanisms — Part 1: General model. September 1991.
2. ISO/IEC 9798-3: 1993. Information technology — Security techniques — Entity authentication mechanisms — Part 3: Entity authentication using a public key algorithm. November 1993.
3. ISO/IEC 9798-2: 1994. Information technology — Security techniques — Entity authentication — Part 2: Mechanisms using symmetric encipherment algorithms. December 1994.
4. ISO/IEC 9798-4: 1995. Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function. March 1995.
5. ISO/IEC 2nd CD 9798-5: 1995. Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero knowledge techniques. June 1996.
6. M.J. Beller, L. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE Journal on Selected Areas in Communications*, 11:821–829, 1993.
7. S.M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. *Computer Communication Review*, 20(5):119–132, October 1990.

8. S. Bengio, G. Brassard, Y.G. Desmedt, C. Goutier, and J. Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, 4:175–183, 1991.
9. L. Chen, D. Gollmann, and C. Mitchell. Key distribution without individual trusted authentication servers. In *Proceedings: the 8th IEEE Computer Security Foundations Workshop*, pages 30–36. IEEE Computer Society Press, Los Alamitos, California, June 1995.
10. I. Damgard. Towards practical public key systems secure against chosen ciphertext attacks. *Lecture Notes in Computer Science 576, Advances in Cryptology - CRYPTO '91*, pages 445–456, 1991.
11. L. Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in Communications*, 11:657–662, 1993.
12. L. Gong. Variations on the themes of message freshness and replay. In *Proceedings: the Computer Security Foundations Workshop VI*, pages 131–136. IEEE Computer Society Press, Los Alamitos, California, June 1993.
13. ETSI/PT12 GSM-03.20. Security related network functions. August 1992.
14. B. Klein, M. Otten, and T. Beth. Conference key distribution protocols in distributed systems. In P. G. Farrell, editor, *Codes and Cyphers, Proceedings of the Fourth IMA Conference on Cryptography and Coding*, pages 225–241. Formara Limited. Southend-on-sea. Essex, 1995.
15. K-Y. Lam. Building an authentication service for distributed systems. *Journal of Computer Security*, 2:73–84, 1993.
16. K.Y. Lam and T. Beth. Timely authentication in distributed systems. In *Lecture Notes in Computer Science 648, Advances in European Symposium on Research in Computer Security*, pages 293–303. Springer-Verlag, 1992.
17. C. Mitchell. Security in future mobile networks. The 2nd International Workshop on Mobile Multi-Media Communications (MoMuC-2), Bristol University, April 11th-13th 1995.
18. C. Mitchell and A. Thomas. Standardising authentication protocols based on public key techniques. *Journal of Computer Security*, 2:23–36, 1993.
19. M. Reiter, K. Birman, and R. van Renesse. Fault-tolerant key distribution. Technical Report 93-1325, Department of Computer Science, Cornell University, Ithaca, New York, January 1993.
20. B. Simons, J.L. Welch, and N. Lynch. An overview of clock synchronization. In *Lecture Notes in Computer Science 448, Advances in Fault-Tolerant Distributed Computing*. Springer-Verlag, 1990.
21. R. Yahalom, B. Klein, and T. Beth. Trust-based navigation in distributed systems. European Institute for System Security, Karlsruhe University, Technical Report 93/4, 1993.

This article was processed using the LaTeX macro package with LLNCS style