Standardising authentication protocols based on public key techniques

Chris Mitchell and Andy Thomas

Tel.: +44–784–443423
Fax: +44–784–439786
Email: cjm@dcs.rhbnc.ac.uk

Technical Report

CSD-TR-92-32

12th July 1993 (revised version)

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

Abstract

ISO has been working on a multi-part authentication mechanisms standard for some years. The first part, ISO/IEC 9798–1 [15], has recently been published. Parts two and three (9798–2, [18] and 9798–3, [17]), covering authentication mechanisms based on symmetric and asymmetric cryptographic techniques respectively, are now moving towards DIS (Draft International Standard) and full International Standard status respectively. This paper is concerned with authentication mechanisms based on asymmetric cryptography; more specifically it contrasts two important authentication mechanisms from the latest version of 9798–3 and from CCITT Recommendation X.509-1988 [9], and briefly illustrates certain known attacks against this type of mechanism. A new potential security problem is then described, to which many published mechanisms appear to be prone. Possible solutions to this problem are discussed, together with potential ramifications on existing standardisation activity.

1 Introduction

This paper is concerned with mechanisms which provide mutual proof of identity and timeliness to a pair of communicating entities, based on the use of digital signatures. Note that these mechanisms consist of an exchange of messages between the two parties. Hence, what we call here *mechanisms* are often referred to as *authentication protocols*; however, for conformance with ISO usage we avoid the use of the word protocol, which has a specific and different meaning in the context of the OSI reference model.

There is a well-established need for standardised authentication mechanisms for a variety of computer networking applications. The need for such mechanisms was recognised as long ago as 1978 by Needham and Schroeder, [30], who gave a variety of different mechanisms based on both symmetric and asymmetric cryptography. Many of the mechanisms which have been standardised or are close to standardisation are derived from examples to be found in [30] and in contemporary work, such as that of Popek and Kline, [31] and Smid, [32]. In parallel with these developments, a need has also long been recognised for key distribution over insecure communication paths; mechanisms for key management have been devised in the context of communications security devices, and often closely resemble authentication mechanisms. Indeed, as is well known, the goals of authentication and key distribution mechanisms are often virtually identical, although the term *key distribution* arises from communications security applications and the term *authentication* arises from the particular security needs of networked computers.

Example applications of authentication mechanisms include:

◇ the CCITT X.400 electronic mail recommendations, in particular X.411-1988 [8], which provide for peer entity authentication between message handling entities prior to the exchange of X.400 messages, based on the use of digital signatures and public key encryption,

◇ the Kerberos authentication service, [34], (part of MIT's Project Athena) which uses the symmetric block cipher DES, [1, 29] and an on-line authentication server to provide mutual authentication between communicating computer applications, and

◇ the DEC proprietary SPX authentication service, [37], which uses public key certificates to provide authentication within a distributed system.

Note that both X.400 and SPX are based on the use of the CCITT X.509 'Directory authentication framework', [9], which standardises a format for public key certificates and also provides three authentication mechanisms based on the use of digital signature schemes.

For several years ISO/IEC JTC1/SC27/WG2, and its predecessors ISO TC98/SC20/WG1 and WG2, have been working on a multi-part authentication mechanisms standard. To date the following progress has been achieved:

◇ Part 1, the 'General Model' was recently published as an international standard: ISO/IEC 9798-1, [15],

◇ Part 2, covering authentication mechanisms based on the use of symmetric cryptography, was recently subject to a ballot as an ISO/IEC *Committee Draft (CD)*[1], [18], and

- ◇ Part 3, concerned with authentication mechanisms based on the use of asymmetric cryptography (i.e. primarily on digital signature mechanisms) has recently received a favourable vote as a DIS, and will therefore be published as a full International Standard in the next year or so, [17].

In this paper we are primarily concerned with authentication mechanisms of the type covered in part 3 of the authentication mechanisms standard, namely those based on the use of digital signatures. We focus on one particular type of mechanism, which uses *nonces* (defined below) to achieve mutual authentication, i.e. authentication of both of a pair of communicating parties to one another.

2 Notation and assumptions

We assume throughout this paper that there exist a pair of parties denoted A and B, which wish to check each other's identity by exchanging messages over an untrusted communications path. As a result of this message exchange they will also typically exchange secret *session key(s)*, to be used for securing subsequent exchanges of data. Whilst we observe how such keys may be exchanged within the context of the mechanisms considered here, this is not the main focus of this paper.

We assume that A and B are both equipped with public key/secret key pairs for the same digital signature algorithm. Following [25], A will therefore have a pair of transformations, denoted S_A (for 'signing') and V_A (for 'verifying'), used for computing signatures for A and verifying the signatures of A respectively. A keeps S_A secret and publicises V_A. We write $S_A(X)$ for the result of applying A's secret signing transformation to data string X and $V_A(X, S)$ for the result of applying A's public verification function to the pair (X, S), where S purports to be A's signature on data X. The result of applying V_A will be either *True* or *False*. B will be similarly equipped with functions S_B and V_B.

We also assume that this signature scheme is operated in conjunction with a collision-free hash function h, so that if A wishes to sign data X, then what is actually computed is $S_A(X) = S'_A(h(X))$, where S'_A is the actual digital signature transformation (e.g. in the case of RSA digital signatures, S'_A represents modular exponentiation using A's secret exponent). Hence, in particular, we are assuming that X is not recoverable from $S_A(X)$. The verification transformation V_A on a pair (X, S) will then typically consist of checking that $h(X)$ is equal to $V'_A(S)$, where V'_A is the (public) inverse of S'_A (e.g. for RSA digital signatures, V'_A represents modular exponentiation using A's public exponent).

Finally observe that we assume throughout that B is equipped with a trusted copy of A's public signature key and vice versa. This may be achieved by use of public key certificates signed by one or more trusted third parties, or by any other reliable means of public key exchange. For further discussion of digital signatures, hash functions and public key certificates see, for example, [25].

[1] CD status is the first of two stages of official draft through which all, or almost all, ISO standards have to pass; the second stage is the *Draft International Standard (DIS)*.

3 Time stamps, sequence numbers and nonces

In any authentication mechanism, be it based on the use of symmetric or asymmetric cryptography, there needs to be a means for checking the *timeliness* of the exchanged messages. By timeliness (or *freshness*) we mean the property that a message has just been generated and sent, rather than being a replay of a message sent at some previous time. Observe that giving a completely precise definition of timeliness can cause some semantic difficulties, since different means of guaranteeing it often give timeliness properties with subtly distinct meanings. However we do not consider this point further here since the informal notion of timeliness given above is all we need in this discussion. For further discussions of timeliness and its provision see, for example, [2, 12, 20].

The timeliness guarantee is necessary to prevent a would-be intruder from impersonating A by replaying intercepted valid authentication messages signed by A. There are three commonly accepted devices used to help guarantee timeliness:

- ◇ Time stamps,
- ◇ Sequence numbers,
- ◇ Challenge-response *nonces*.

The latest drafts of 9798–2 and 9798–3 both include examples of authentication mechanisms based on the use of all three methods of guaranteeing freshness. However, this paper is concerned primarily with authentication mechanisms based on the use of nonces, and we do not discuss time stamp and/or sequence number based methods further here. Before proceeding, we briefly consider the general use of and requirements for nonces.

The term *nonce* was introduced by Needham and Schroeder, [30]. Protocols based on nonces have the disadvantage relative to time stamps or sequence numbers that they typically require an extra message in the authentication exchange. However, their use avoids the need for synchronised clocks and/or the storage of sequence numbers for every party with which an authentication exchange may be required.

A nonce (or *challenge*) is a value included in a message whose inclusion in a subsequent response can guarantee the freshness of that response. To be effective it is necessary that a nonce is only ever used once by any one party during the lifetime of that party's key. There are two ways commonly used for ensuring the 'one-time' property of nonces. Firstly, if nonces are always chosen at random, then, given that each nonce contains a sufficiently large number of bits, the probability of nonce repetition can be made very close to zero. Alternatively, each entity can be equipped with a single counter, from which nonces are extracted as required (with the counter being incremented every time a nonce is used).

Because of the practical difficulties involved in generating truly random numbers, generating nonces using a counter, or a pseudo-random source, may appear attractive. One possible problem with the use of a counter is that, if an entity loses state, then a problem arises when the counter has to be initialised. Of course if a new key pair is generated simultaneously then there is no problem, but that will not always be the case—in some applications public key/secret key pairs for digital signature algorithms are likely to remain in use for considerable periods of time.

Perhaps more seriously, counters, and other non-random methods for generating nonces, can give rise to 'preplay' attacks. In such attacks, a third party may evince a response to a predicted nonce ahead of time, and play it back as an untimely response when the same nonce is genuinely sent. To avoid this requires either

1. using only unpredictable nonces, or
2. providing cryptographic measures to guarantee the origin of every message of an authentication exchange.

Because the second solution is only appropriate to some mechanisms, it is typically automatically required of nonces that they be unpredictable (hence ruling out counter-generated nonces), but this is not always strictly necessary.

4 Two example mechanisms

We now consider two examples of authentication mechanisms, both of which use nonces and digital signatures to provide mutual authentication. The first, which we call mechanism Q, is a simplified version of the 'three-way authentication' mechanism specified in CCITT Recommendation X.509-1988, [9]. The second, which we call mechanism R, is taken directly from Clause 5.2.2 of the latest version of 9798–3, [17], where it is referred to as the 'three pass authentication' mechanism.

Mechanism Q

Q1. **A → B:** $R_A, D_1, S_A(R_A, B, D_1)$

Q2. **B → A:** $R_B, D_2, S_B(R_A, R_B, A, D_2)$

Q3. **A → B:** $S_A(R_B)$

We now briefly describe the procedural aspect of this mechanism. Note that, as in any authentication mechanism, the recipient of each message performs a check, and only proceeds with the mechanism if the check proceeds correctly.

On receipt of message $Q1$:

> B checks A's signature on the string R_A, B, D_1, where R_A and D_1 are recovered from the unsigned portion of the message and B is simply the name of B (which B is assumed to know).

On receipt of message $Q2$:

> A checks B's signature on the string R_A, R_B, A, D_2, where R_B and D_2 are recovered from the unsigned portion of the message, and R_A has been stored by A.

On receipt of message $Q3$:

> B checks A's signature on the string R_B, where R_B has been stored by B.

The purpose of this mechanism is that, having performed this exchange of messages, A and B should now be sure that they are talking to one another, and that the data strings D_1 and D_2, which might for example include encrypted keys, originate from one another. Moreover they should also be sure that all the exchanged messages were fresh. Unfortunately, as we see below, this mechanism is flawed, and hence A and B cannot be sure that the mechanism has not been manipulated by third parties.

Mechanism R

R1. **A → B:** R_A

R2. **B → A:** $R_B, D_1, S_B(R_B, R_A, A, D_1')$

R3. **A → B:** $D_2, S_A(R_A, R_B, B, D_2')$

The procedural aspects of this mechanism are somewhat similar to those for mechanism Q. Note that one important difference between the two mechanisms is that, in mechanism R, the signed data strings (D_1' and D_2') are different from the data strings sent unsigned (D_1 and D_2). We explain the use of these data strings further in Section 4.1 below.

In both of these two examples R_A and R_B are nonces, chosen by A and B respectively. In the standards documents R_A and R_B are referred to as random numbers. However, as we have already discussed, they need not be genuinely random but can be pseudo-random; the only vital properties that they must possess are

- ◇ the 'one-time' property, i.e. A must never choose the same R_A twice, at least not within the lifetime of a signature key pair, and
- ◇ unpredictability (although, since all three messages of mechanism Q are signed, this property is not strictly necessary for the first of the two mechanisms).

We now consider three aspects of the design of these two mechanisms, and in doing so describe a number of attacks that have been devised against these and other similar authentication mechanisms.

4.1 The role of data strings

The roles of the data strings D_1, D_2 in mechanism Q and of D_1, D_1', D_2, D_2' in mechanism R merit some explanation. These strings have been provided to enable secure key exchange to be achieved simultaneously with mutual authentication. In addition, in some circumstances these strings can also be used to convey other data items, such as an authentication check for an accompanying message, the origin of which can be verified since the data strings are signed. It is clear that means must be provided for conveying the value of these data strings from source to recipient, so they are included in 'plain text' as well as within the scope of the signature. It is at this point that the question naturally arises as to why mechanism R provides for different data strings D_1, D_1' and D_2, D_2'.

This has been done to counter a serious security problem, first identified by Burrows, Abadi and Needham, [7], that occurs in certain applications of mechanism Q. Essentially, a problem arises when the data string D_1 of the first message of the mechanism (i.e. $Q1$) is used to contain both an encrypted key, K say, and an integrity check for data encrypted using K. The recipient of such a data string is then unable to determine whether or not the originator of the message actually knew the unencrypted data, or simply hijacked an encrypted key and corresponding encrypted data from a third party's message. This problem arises naturally in X.400 electronic mail, and scenarios in which this type of misuse of third party data causes significant security problems are described in more detail in [14, 26, 27].

This problem can be solved by signing an unencrypted version of the key K, but sending K encrypted (K might, for example, be encrypted using the public key of the intended recipient). However, this then means that the data string which is signed is different from the plain text data string, and this explains why message $R2$ includes D_1 and D'_1. Of course, in order for the signature to be checked by the legitimate recipient, it is necessary for D'_1 to be reconstructable using some combination of information contained within D_1 and other information held by the legitimate recipient. Note that signing unencrypted keys does not risk compromise of their secrecy, since we assume that all data strings are subject to a one-way hash function prior to application of the signature operation.

4.2 Another Burrows–Abadi–Needham attack

In addition to this problem with data strings, Burrows, Abadi and Needham, [7], found a more fundamental problem with mechanism Q. Suppose A and B have used mechanism Q on some previous occasion, and that malicious user C has intercepted the three messages $Q1, Q2, Q3$. Now suppose that C wishes to impersonate A to B. We suppose, for the sake of simplicity, that the data strings D_1 and D_2 are null and omit them from this description.

C initially sends the first of the three previously intercepted messages to B:

Q1. **C → B:** $R_A, S_A(R_A, B)$

B responds (thinking it is talking to A, but actually talking to C). It challenges C with a new nonce, $R_{B'}$.

Q2. **B → C:** $R'_B, S_B(R_A, R'_B, A)$

C meanwhile causes A to initiate authentication with C (by some means). As a result A sends C the following message:

Q1. **A → C:** $R'_A, S_A(R'_A, C)$

C, when responding to A, uses the random value R'_B, provided to C by B:

Q2. **C → A:** $R'_B, S_C(R'_A, R'_B, A)$

A responds with the following message

Q3. **A → C:** $S_A(R'_B)$

But this is exactly what C needs to convince B (falsely) that it is talking to A, i.e. C can now send

Q3. **C → B:** $S_A(R'_B)$

and B will believe that it is talking to A whereas it is actually talking to C.

The above discussion is a paraphrased version of Burrows, Abadi and Needham's text. Their solution is a very simple one, and merely requires a small change to the third message of the mechanism to the following.

Q3′. **A → B:** $S_A(R_B, B)$

Thus the only change necessary to avoid the problem is to include the name of B in the third message of the mechanism.

4.3 A Canadian attack

Having considered two major flaws in mechanism Q, and indicated their influence on the design of mechanism R, we conclude this discussion of these two example mechanisms by examining one further aspect of mechanism R. It is by no means immediately obvious why R_A is included in the signed part of $R3$, although its presence is vital, as we see below. Indeed, in the immediately preceding version of 9798–3, R_A was not included in $R3$, and the mechanism had the following, slightly different, form:

R1. **A → B:** R_A

R2. **B → A:** $R_B, D_1, S_B(R_B, R_A, A, D'_1)$

R3′. **A → B:** $R'_A, D_2, S_A(R'_A, R_B, B, D'_2)$

This version of the mechanism is subject to an attack put forward to ISO by the Canadian Member Body in 1991, [16]. In this attack, an intruder, C say, convinces A to produce (as message $R2$ of the mechanism) exactly what C needs to impersonate A to B as the third message of the mechanism. In more detail the attack operates as follows.

C first chooses a random R_A and calls B (pretending to be A):

R1. **C → B:** R_A

B responds to C (thinking it is responding to A):

R2. **B → C:** $R_B, D_1, S_B(R_B, R_A, A, D'_1)$

C now calls A pretending to be B, invoking a second copy of the mechanism. C uses the nonce R_B just chosen by B:

R1. **C → A:** R_B

A responds to C (thinking it is responding to B):

R2. **A → C:** $R'_A, D''_1, S_A(R'_A, R_B, B, D'''_1)$

C now takes this response from A and forwards it directly to B as the third message of the mechanism:

R3′. **C → B:** $R'_A, D''_1, S_A(R'_A, R_B, B, D'''_1)$

B now believes itself to be talking to A, although it is actually talking to C.

Note that this attack on the 'old' version of mechanism R was only possible because of the presence of the random value R'_A in the third message of the mechanism. The presence of this value in message $R3$ makes the formats of messages $R2$ and $R3$ identical. This is what enables C to use the second message of one instance of the mechanism as the third message of another instance of the mechanism.

As a result one might argue that the removal of R'_A from the third message of the mechanism is sufficient to prevent the attack and hence the addition of R_A is unnecessary. However it is not clear that, to comply with the standard, implementors will be obliged to sign a string of data items in the same order as specified in the standard. Given the freedom which implementors will also have over the format and length of data strings, the inclusion of R_A in the third message is still necessary to prevent attacks against some possible mechanism implementations. Given the necessity to standardise mechanisms which are robust in all types of use, the inclusion of R_A appears to be a sound precaution.

The random number R'_A was actually inserted into the third message of the mechanism for a completely unrelated function, namely to protect A against attacks which might be possible if the cryptographic hash function used to construct signatures is not collision-free, a property which is normally desirable for hash functions used in constructing digital signatures (see, for example, [25]). Consider the string which A has to sign in order to construct message $R3'$ in normal operation of the mechanism. If R'_A is not present then this string is completely known in advance to B. In such a circumstance B could, by manipulating the choice of R_B and using the weakness of the hash function, construct a string $X = R_B, B, D'_2$ whose hash value $h(X)$ is the same as the hash value of some message M which B would like A to sign (and which A would not like to sign). That is, given the hash function is not collision-free, B might be capable of finding a nonce R_B for which

$$h(X) = h(M)$$

where $X = R_B, B, D'_2$.

It might be argued that a standardised authentication mechanism should not incorporate features designed to protect users against shortcomings of cryptographic techniques. Indeed, this reasoning appears to have been followed by the designers of the later mechanism R, which does not protect against such attacks. Observe that mechanism R could very simply be modified to protect against this 'weak hash' attack by changing message $R3$ to:

R3″. **A → B:** $R'_A, D_2, S_A(R'_A, R_A, R_B, B, D'_2)$

Finally note that, in order for the use of a random value R'_A to be effective in protecting against 'weak hash' attacks, it requires properties different from those which one would require of a nonce. First and foremost the value of R'_A must be *unpredictable*, i.e. party B must not be able to predict what value A will choose for R'_A. This means that a genuinely random or cryptographically strong pseudo-random number generator must be available to the generator of R'_A. In addition, for B to be similarly protected against a similar attack by A, the nonce R_B must also be generated in the same way.

We do not consider the effects of weak cryptographic mechanisms further here.

5 A possible new problem and a solution

Both mechanisms Q and R, and all their variants, require both A and B to be prepared to sign data strings X of the form

$$X = \ldots, R, \ldots$$

where R is a nonce supplied by another party. In certain situations this can cause significant security problems, as we now describe.

Suppose A has a single secret key/public key pair, used for a number of different digital signature applications, such as:

- ◇ digital signatures for authentication mechanisms,
- ◇ digital signatures for message origin authentication, message integrity and/or message non-repudiation,
- ◇ digital signatures for software integrity checking.

By signing a string of the form $X = \ldots, R, \ldots$ as part of an authentication mechanism, A allows the supplier of R (say B) to choose part of the string X which A signs. By careful selection of R, B might be able to arrange for X to resemble a message with meaning chosen by B. Once A signs X, B is then in possession of A's signature on a message which A never had any intention of signing, and which B may be able to use for malicious purposes. Hence one message of an authentication mechanism could be misused as a signed string in some other application.

One 'obvious' solution is to use a different signature/hash function combination and/or public key/secret key pair for each application where digital signatures are required. Although possible in some circumstances, this is undesirable as a universal solution for the following reasons.

- ◇ It is convenient to use the same digital signature/hash function pair for each application where digital signatures are required—hence the only possibility is to vary the key for each application.
- ◇ Whilst the use of a different key for each application would solve the problem, it is not always convenient or even possible to follow such a route. In the highly integrated computer networks of the future there may be a large variety of inter-related networked applications all requiring the use of digital signatures. Management of a large number of key pairs, one for each application, could become a serious problem, particularly if they are to be stored in a low-cost user device with limited memory capacity. Moreover, if the X.509 key certification scheme is to be used to disseminate user public keys (as is already proposed within X.400 electronic mail and the SPX authentication service), then problems arise because there is no field within an X.509 certificate for marking the *scope* of a key, i.e. marking the application(s) for which this key is to be used. This limitation of X.509 certificates has been pointed out previously in a completely different context (see, for example, Section 5.2 of [26]) and modification of the X.509 Recommendation to provide for such a field would be a great advantage.

It should be pointed out that so called ‘key separation’ (i.e. the use of different keys for different applications) is already widespread in applications of conventional (symmetric) cryptography. For example, DES is routinely used with key separation both ‘vertically’ (in a multi-level hierarchy with *key encrypting keys* and *working keys*) and ‘horizontally’ (with different working keys for different functions). One means of achieving this key separation has been to associate bit strings (‘control vectors’) with DES keys to limit their application. This type of approach may also become widely used in applications of public key cryptography, although, because of the types of problem mentioned above (in particular the problem arising from the shortcomings of X.509 key certificates) alternative solutions do need to be examined seriously.

The alternative solution we consider here is always to include an application/mechanism identifier in the signed string; this identifier must be in a standardised form and in a fixed location in the string (e.g. always at the beginning). This sets the ‘context’ in which the signed string should be interpreted, and is, in some sense, the most logical solution. It seems reasonable to insist that one should always include information as to the context, and thus to the meaning, of a string before putting one’s signature to it. Indeed, if one considers the conventional analogy with a human signature, one would never reasonably expect anyone to sign a document without some reasonable understanding of its meaning and context (although this does seem to be the norm for insurance and time-share salesmen!).

An additional advantage of the identifier approach is that, if the identifier also includes the number of the message in the authentication mechanism, then certain types of attack may be ruled out—this in turn may enable the mechanism itself to be simplified. In particular the Canadian Member Body attack on the earlier version of mechanism R appears to be prevented by the inclusion of such an identifier, and, as a result, mechanism R may be simplified to the following.

R1. **A → B:** R_A

R2′. **B → A:** $R_B, D_1, S_B(ID_2, R_B, R_A, A, D_1')$

R3‴. **A → B:** $D_2, S_A(ID_3, R_B, B, D_2')$

Of course, this version of mechanism R needs considerable further checking before it could be recommended for adoption as part of 9798–3. Note that, by similar reasoning to that used to enable R_A to be omitted from the signed part of message $R3'''$, it may also be possible to omit R_B from the signed part of message $R2'$—this possibility also needs careful checking.

Before proceeding, observe that the problem of the context of a signature applies to all uses of digital signatures, not just to their use in authentication mechanisms. It would therefore seem sensible to suggest the adoption of application identifiers in all standardised uses of digital signatures. However, the need to sign material supplied by another entity seems most likely to arise in the context of peer entity authentication based on nonces, and hence it is not entirely illogical to raise the problem in this context.

6 Future directions for research

It is not clear how carefully the five existing mechanisms in the current draft of 9798–3 have been checked using the available tools. In particular it would seem to be judicious to apply the known logical tools for checking authentication mechanisms to all five mechanisms in 9798–3. This work should include application of the logics of

- ◇ Burrows, Abadi & Needham (1988–90), [4, 5, 6, 7], and various extensions by Gaarder & Snekkenes (1990/91), [10, 33], Kailar & Gligor (1991), [19], and Gong, Needham & Yahalom (1990/91), [11, 13],
- ◇ Meadows (1989–91), [21, 22, 23, 24],
- ◇ Syverson (1990/91), [35, 36], and
- ◇ Bieber (1990), [3].

Work also needs to be performed on applying these authentication logics to versions of the existing 9798–3 mechanisms when they have been modified to incorporate the use of mechanism/application identifiers. This may require the modification of existing logics to recognise the value of such identifiers, in particular where they label the stage of the mechanism a signed value represents.

More generally the adoption of application/mechanism identifiers needs to be pursued on a broad front, wherever the use of digital signatures is being standardised. In the short term this can be done by attempting to influence the emerging work on digital signature standardisation (e.g. NIST's proposed Digital Signature Standard (DSS), [28], and corresponding work in ISO which is at an early stage). In the longer term, the work on standardising the form of *Security Information Objects* being pursued within ISO/IEC JTC1/SC27/WG1 is the most logical place to promote the general use of standardised application/mechanism identifiers—however, this work may take a number of years to mature.

In short, much needs to be done if agreed and secure authentication mechanisms are to be standardised in the short term. It is still to be hoped that a DIS version of 9798–3 can be agreed within the next twelve months, but many hurdles still have to be crossed.

Acknowledgements

The authors would like to thank Dieter Gollmann, Kwok-Yan Lam and John Leach for their many valuable comments on, and suggestions for improvement to, earlier drafts of this paper. The second author would also like to thank Hewlett-Packard Ltd. for their support under SERC CASE studentship No. 90700562.

References

[1] American National Standards Institute, New York. *ANSI X3.92–1981, Data Encryption Algorithm*, 1981.

[2] R.K. Bauer, T.A. Berson, and R.J. Feiertag. A key distribution protocol using event markers. *ACM Transactions on Computer Systems*, **1**(3):249–255, 1983.

[3] P. Bieber. A logic of communication in hostile environment. In *Proceedings: The Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, Los Alamitos, California, June 1990.

[4] M. Burrows, M. Abadi, and R. Needham. Authentication: A practical study in belief and action. In M. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning and Knowledge*. Morgan Kaufmann, Los Altos, California, 1988.

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, Arizona, December 3–6, 1989*. ACM, 1989.

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London, Series A*, **426**:233–271, 1989.

[7] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report 39, Digital Equipment Corporation Systems Research Center, Palo Alto, California, February 1990. Revised version.

[8] Comité Consultatif International de Télégraphique et Téléphonique. *CCITT Recommendation X.411–1988, Message Handling Systems: Message Transfer System: Abstract Service Definition and Procedures*, 1988.

[9] Comité Consultatif International de Télégraphique et Téléphonique. *CCITT Recommendation X.509–1988, The directory—Authentication framework*, 1988.

[10] K. Gaarder and E. Snekkenes. On the formal analysis of PKCS authentication protocols. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology: AUSCRYPT '90*, pages 106–121, Sydney, Australia, 1990. Springer-Verlag, Berlin.

[11] L. Gong. Handling infeasible specifications of cryptographic protocols. In *Proceedings: The Computer Security Foundations Workshop IV*, pages 99–102. IEEE Computer Society Press, Los Alamitos, California, June 1991.

[12] L. Gong. A security risk of depending on synchronised clocks. *ACM Operating Systems Review*, **26**(1):49–53, January 1992.

[13] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings: 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, Los Alamitos, California, 1990.

[14] C. I'Anson and C.J. Mitchell. Security defects in CCITT recommendation X.509 — the directory authentication framework. *ACM Computer Communication Review*, **20**(2):30–34, 1990.

[15] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798-1: 1991, Information Technology—Security techniques—Entity authentication mechanisms—Part 1: General model*, 1991.

[16] International Organization for Standardization, Genève, Switzerland. *ISO/IEC JTC1/SC27 N313 (1991-10-02), Summary of voting on Letter Ballot No. 6, document SC27 N277, CD9798-3.3 "Entity authentication mechanisms, Part 3: Entity authentication using a public key algorithm"*, October 1991.

[17] International Organization for Standardization, Genève, Switzerland. *ISO/IEC JTC1/SC27 N497 (1992-07-06), Revised text of CD 9798-3.4 'Entity authentication mechanisms, Part 3: Entity authentication using a public key algorithm', submitted for DIS processing*, July 1992.

[18] International Organization for Standardization, Genève, Switzerland. *ISO/IEC JTC1/SC27 N584 Rev (1992-11-10), ISO/IEC 4th CD 9798-2, Information technology—Security techniques—Entity authentication mechanisms—Part 2: Entity authentication using symmetric techniques*, November 1992.

[19] R. Kailar and V.D. Gligor. On belief evolution in authentication protocols. In *Proceedings: The Computer Security Foundations Workshop IV*, pages 103–116. IEEE Computer Society Press, Los Alamitos, California, June 1991.

[20] K.-Y. Lam. Building authentication services for distributed systems. *Journal of Computer Security*, to appear.

[21] C. Meadows. Using narrowing in the analysis of key management protocols. In *Proceedings: 1989 IEEE Computer Society Symposium on Security and Privacy*, pages 138–147. IEEE Computer Society Press, Los Alamitos, California, May 1989.

[22] C. Meadows. Representing partial knowledge in an algebraic security model. In *Proceedings: The Computer Security Foundations Workshop III*, pages 23–31. IEEE Computer Society Press, Los Alamitos, California, June 1990.

[23] C. Meadows. A system for the specification and analysis of key management protocols. In *Proceedings: 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 182–195. IEEE Computer Society Press, Los Alamitos, California, 1991.

[24] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, **1**:5–35, 1992.

[25] C.J. Mitchell, F.C. Piper, and P.R. Wild. Digital signatures. In G.J. Simmons, editor, *Contemporary cryptology: The science of information integrity*, pages 325–378. IEEE Press, 1992.

[26] C.J. Mitchell, P.D.C. Rush, and M. Walker. A secure messaging architecture implementing the X.400–1988 security features. *The Computer Journal*, **33**:290–295, 1990.

[27] C.J. Mitchell, M. Walker, and P.D.C. Rush. CCITT/ISO standards for secure message handling. *IEEE Journal on Selected Areas in Communications*, **7**:517–524, 1989.

[28] National Institute of Standards and Technology (NIST), Gaithersburg, MD. *A proposed Federal Information Processing Standard for Digital Signature Standard (DSS)*, August 1991.

[29] National Technical Information Service, Springfield, Va. *National Bureau of Standards (NBS) Federal Information Processing Standards (FIPS) Publication 46—Data Encryption Standard (DES)*, April 1977.

[30] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, **21**:993–999, 1978.

[31] G.J. Popek and C.S. Kline. Encryption and secure computer networks. *Computing Surveys*, **11**:331–356, 1979.

[32] M.E. Smid. Integrating the Data Encryption Standard into computer networks. *IEEE Transactions on Communications*, **COM-29**:762–772, 1981.

[33] E. Snekkenes. Exploring the BAN approach to protocol analysis. In *Proceedings: 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 171–181. IEEE Computer Society Press, Los Alamitos, California, May 1991.

[34] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: an authentication service for open network systems. In *Proceedings: Usenix Association, Winter Conference, Dallas 1988*, pages 191–202. USENIX Association, Berkeley, California, February 1988.

[35] P. Syverson. Formal semantics for logics of cryptographic protocols. In *Proceedings: The Computer Security Foundations Workshop III*, pages 32–41. IEEE Computer Society Press, Los Alamitos, California, June 1990.

[36] P. Syverson. The use of logic in the analysis of cryptographic protocols. In *Proceedings: 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 156–170. IEEE Computer Society Press, Los Alamitos, California, May 1991.

[37] J.J. Tardo and K. Alagappan. SPX: Global authentication using public key certificates. In *Proceedings: 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 232–244. IEEE Computer Society Press, Los Alamitos, California, May 1991.