Security Defects in CCITT Recommendation X.509 - The Directory Authentication Framework

Colin I'Anson
Hewlett Packard Laboratories,
Filton Rd, Stoke Gifford,
Bristol, BS12 6QX

Chris Mitchell
Computer Science Department
RHBNC,
Egham Hill, Egham
Surrey TW20 0EX

Abstract

Security defects to be found in C.C.I.T.T. Recommendation X.509-1988 are described together with possible solutions.

1. Introduction

The purpose of this short paper is to discuss certain security defects which have recently been discovered in C.C.I.T.T. Recommendation X.509-1988, [2]. Each defect is described and a possible solution is outlined. A discussion of the use of the X.509 authentication framework to provide security for X.400 electronic mail may be found in [4]; this latter paper also briefly discusses security defects to be found in the X.400 Recommendations.

2. Over-restrictive definition of digital signatures

2.1. Problem description

The Authentication Framework described in CCITT Recommendation X.509 allows both generic public key cryptosystems and generic symmetric cryptosystems. However the technique defined in clause 8.1 of X.509 for producing digital signatures requires the underlying public key cryptosystem to have very special properties (outlined in clause 6.1 of X.509). In practice this almost certainly mandates the use of the RSA public key cryptosystem, described in Annex C of X.509, and rules out use of other, potentially more desirable, digital signature techniques.

The root cause of the problem would seem to be the desire for each user to have a single public key/secret key pair, used for both the generation of digital signatures and the encryption of data (for confidentiality purposes). Note that the requirement for both these types of cryptographic operation are present in all three of the authentication protocols described in clauses 9.2-9.4 of X.509.

The observed problems then stem from the need for means to be provided for a single cryptosystem to provide both these types of functionality. In X.509 it is assumed that there exists a single public key cryptosystem which is used in two ways:

- if user A wishes to provide confidentiality for data sent to user B then A encrypts this data using B's public key.

- if user A wishes to provide authentication services (e.g. origin authentication and integrity) for data sent to user B then A decrypts this data using A's secret key.

As described in clause 6.1, this requires the public key cryptosystem used to have *the property that both keys in the key pair can be used for encipherment.* In practice this virtually mandates the use of the RSA cryptosystem.

2.2. Possible solutions

There are two alternatives to the X.509 scheme, both of which remove the need for such restrictive requirements on the cryptosystem used. We now describe these two solutions, and note their advantages and disadvantages; either solution could be included as an enhancement to the Recommendation to remove the restrictive requirements.

The first alternative stems from the observation that X.509 does not need to specify so closely how digital signatures should be derived from the underlying public key cryptosystem (no more than it needs to specify which cryptosystem or hash function is used). All that is needed is some means of indicating (probably by *algorithm-identifiers*) which means is in use. Many public-key cryptosystems can be used to create digital signatures, but the means by which this is done tends to be specific to the algorithm concerned.

The main advantage of this alternative is that it requires a minimal number of changes to X.509; in particular it retains the idea of using a single key pair for both data encryption and digital signature. Moreover, the ASN.1 SIGNED and SIGNATURE macros defined in clauses 8.5 and 8.6 remain perfectly satisfactory as they are currently defined, since they provide an algorithm-identifier for indicating which signature algorithm is in use. This identifier can be used to both specify the underlying public key cryptosystem and the way in which it is to be used to provide a digital signature.

The main disadvantage with this alternative is that it does not allow use of digital signature schemes not derived directly from public key cryptosystems. This is unfortunate since some of the most promising recent suggestions for digital signature schemes are based on the theory of zero- knowledge protocols, and they do not have corresponding public-key cryptosystems.

The second alternative involves dispensing with the requirement that a single key pair should be used for both data encryption and digital signature. There is no obvious reason why every user should not have two key pairs - one for encrypting and decrypting data and the other for generating and checking signatures.

The main advantage of this alternative is that it allows the use of arbitrary public key cryptosystems and digital signature algorithms. This is particularly important if the scheme is to be useful in the future, when, as is the case at the moment, new algorithms are invented (and broken!) on an almost daily basis.

The main disadvantage with this alternative is that it requires rather radical changes to the existing text of X.509.

3. Potentially defective token structure

3.1. Problem description

The token used in clauses 9.2-9.4 has the following general form (when sent from user A to user B):

$$A\{\ t^A, r^A, B, r^B, \text{sgnData}, Bp[\text{encData}]\ \}$$

where t^A, r^A and r^B are time stamps and random numbers, B is the name of B, sgnData is a collection of non-confidential parameters (e.g. authentication checks for accompanying data), encData is a collection of confidential parameters (e.g. secret keys), Bp[x] denotes the encryption of data x using the public key of B and A{ y } denotes data y with a signed version of y appended. Inclusion of sgnData and encData within the token is intended to provide origin authentication, integrity and non-repudiation services for these parameters.

The problem with this form of token is that it involves signing encrypted data, which is generally accepted to be bad practice. Not only is this undesirable from an aesthetic view-point, but it also leads to potentially serious security loop-holes. We next present an example of how this form of token can lead to a catastrophic security failure.

Suppose user A wishes to make an enquiry of a public database B. The information in the database is public, but the nature of the enquiry is confidential. User A therefore provides confidentiality and authentication checks for the message. We suppose that the message transfer medium between A and B is X.400 electronic mail, the 1988 version of which uses the X.509 authentication token to provide security services for mail items.

User A encrypts the message using some conventional (symmetric) encryption algorithm, and includes the key used in the encData parameter of a token which accompanies the message (the form of the

token is as given above). A also computes an integrity check for the message, and includes this check in the sgnData parameter of the token (if the method used to compute the integrity check is key-based, then any secret key used can be included in the encData portion of the token).

A would then reasonably expect this message to remain confidential. Unfortunately this is not so, as we now explain.

Suppose malicious user C intercepts the message between A and B. C constructs a new message, containing the same encrypted content as the old message, but accompanied by a new token with the form:

$$C\{ t^C, r^C, B, r^B, \text{sgnData}, \text{Bp[encData]} \}$$

where the values sgnData and Bp[encData] are taken directly from the token (generated by A) which accompanied the original message. Note that at this point C cannot read any of the content of the message because C does not know how to decrypt Bp[encData].

When B receives the new message he decrypts it and authenticates it and believes it to come from C. He then constructs an appropriate reply (perhaps including the request) and sends it back to C. C now reads the response and can work out what A's original request was, even though it was encrypted and C did not have the means to decrypt it!

This problem was discovered by Burrows, Abadi and Needham. A brief discussion can be found in Section 11 of [1].

3.2. Possible solution

The most straightforward solution (and one which provides for a certain degree of backwards compatibility) is to allow the use of more than one kind of token. This would require the use of a token-type identifiers, and recognition of the possibility of other kinds of token in the text in X.509. As with the proposed solutions to the previous problem, this could be added to the Recommendation as an enhancement.

If an additional token form is required which does not have the problem identified in the existing token, then the simplest possible solution is probably as follows. This modification involves no additional effort as far as token construction is concerned, and it is simply to require that the encryption of encData is done *after* the signature operation instead of before. This requires the general form of token (sent from A to B) to be as follows:

$$t^A, r^A, B, r^B, \text{sgnData}, \text{Bp[encData]}, \text{As[h(L)]}$$

where

$$L = t^A, r^A, B, r^B, \text{sgnData}, \text{encData}.$$

4. Defective 3-way authentication protocol

4.1. Problem description

The 3-way authentication protocol described in clause 9.4 of X.509 is defective and does not provide a full peer-entity authentication service.

The protocol is as follows:

$$A \rightarrow B: \quad A\{ t^A, r^A, B \}$$

$$B \rightarrow A: \quad B\{ t^B, r^B, A, r^A \}$$

$$A \rightarrow B: \quad B\{ r^B \}.$$

The text of clause 9.4 clearly states that the checking of timestamps t^A and t^B is optional if all three messages are used. Unfortunately, as we now describe, t^A must still be checked if the above protocol is

to work (thus making the third message completely redundant!).

Consider the following example. Suppose A and B have used the above protocol on some previous occasion, and that malicious user C has intercepted the above three messages. In addition suppose that timestamps are not used and are all set to 0. Finally now suppose that C wishes to impersonate A to B.

C initially sends the first of the above three messages to B:

$$C \rightarrow B: \; A\{ \; 0, r^{A}, B \; \}$$

B responds (thinking it is talking to A, but actually talking to C). It challenges C with a new random value, $r^{B'}$:

$$B \rightarrow C: \; B\{ \; 0, r^{B'}, A, r^{A} \; \}$$

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

$$A \rightarrow C: \; A\{ \; 0, r^{A'}, C \; \}.$$

C, when responding to A, uses the random value $r^{B'}$, provided to C by B:

$$C \rightarrow A: \; C\{ \; 0, r^{B'}, A, r^{A'} \; \}$$

A responds with the following message

$$A \rightarrow C: \; A\{ \; r^{B'} \; \}.$$

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

$$C \rightarrow B: \; A\{ \; r^{B'} \; \}$$

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

This problem was discovered by Burrows, Abadi and Needham. The above discussion is a paraphrased version of their text; the proposed solution is also theirs (see [1], Section 11).

4.2. Possible solution

This problem has a very simple fix, namely the inclusion of the name of B in the signed information for the third message. This does, in fact, very much improve the symmetry of the 3- way protocol. The only textual change required in X.509 is to change step 8 of clause 9.4 to:

A sends the following authentication token to B:

$$A\{ \; r^{B}, B \; \}.$$

5. Incorrect conditions on use of RSA in Annex C

5.1. Problem description

Clause C.6.2 of Annex C of X.509 lists properties that RSA keys must satisfy in order to be secure (given current knowledge about the difficulty of factoring large numbers). The clause concludes with the constraint that e and n must satisfy

$$e > \log_2(n).$$

While the constraint is correct, the reason given for requiring it is incorrect.

The relevant text from X.509 is as follows:

> *It must be ensured that $e > log_2(n)$ in order to prevent attack by taking the e'th root mod n to disclose the plaintext.*

Taking the e'th root *modulo* n of a ciphertext block will always reveal the plaintext, no matter what the values of e and n are. In general this is a very difficult problem, and indeed is the reason why RSA is

secure. The point is that, if e is too small, then taking the normal integer e'th root will be the same as taking the e'th root modulo n, and taking integer e'th roots is relatively easy.

5.2. Possible solution

Replace the clause from X.509 quoted above with the following:

> It must be ensured that $e > \log_2(n)$. If not, then the simple operation of taking the integer e'th root of a ciphertext block will disclose the plaintext.

6. Insecure hash function in Annex D

6.1. Problem description

In clause D.2 of Annex D of X.509, a hash function is described which uses arithmetic modulo N. Unfortunately, Coppersmith, [3], has recently shown that this hash function is somewhat flawed. Coppersmith exhibits a method to construct two different messages with the property that the hashed version of one of them is equal to 256 times the hashed version of the other (mod N). In certain circumstances this method can then be used to construct a false signature.

6.2. Possible solution

Replace the clause describing the flawed hash function with text to indicate the example has been deprecated.

7. Use of Security Frameworks

The long term future of Recommendation X.509 is unclear since the material covered is being superseded by the individual security frameworks for authentication, access control, etc., that are being considered by the Distributed Applications Framework (DAF) question in C.C.I.T.T. Study Group VII and associated ISO groups, in conjunction with work on security techniques in ISO/IEC/JTC1/SC27. Until such time as these become Recommendations (or ISO standards), the implementor's guides and information provided by bodies like NIST (National Institute of Standards and Technology) should be utilised.

References

[1] M. Burrows, M. Abadi and R.M. Needham, "A logic of authentication", *ACM Operating Systems Review*, 23 no 5 (1989) 1-13.

[2] C.C.I.T.T. Recommendation X.509, *The Directory - Authentication Framework*, C.C.I.T.T., December 1988.

[3] D. Coppersmith, "Analysis of ISO/CCITT Document X.509 Annex D", IBM Research Division, Yorktown Heights, June 1989.

[4] 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 (1989) 517-524.