Regular Paper

Attacks on Authentication Protocols with Compromised Certificates and How to Fix them

Wu Wen,† Takamichi Saito†† and Fumio Mizoguchi†††

The security of authentication protocols based on public key cryptography depends on the validity and freshness of the certificate. It is usually assumed that a well deployed Public Key Infrastructure (PKI) can guarantee the validity and freshness of certificates through mechanisms such as Certificate Revocation List (CRL) or Online Certificate Status Protocol. In reality, such a guarantee is not always assured. This paper analyzes the security of public key authentication protocols in various situations with compromised certificates. A particular type of attack, namely the "ex-employee attack," against the "named-server, anonymous-client" mode of the SSL/TLS handshake protocol is described, as well as a modified version of the SSL/TLS handshake protocol that can prevent the "ex-employee attack." Methods for analyzing these protocols are also presented.

1. Introduction

To conduct business securely on the Internet, various cryptographic mechanisms and protocols can be used to achieve specific objectives. Two of the most often used methods are authentication and key exchange, as in the Secure Socket Layer (SSL) protocol [7] and its successor, the Transport Layer Security (TLS) protocol [3]. Since the handshake protocols for SSL V3.0 and the TLS V1.0 are identical for this discussion, we will refer to them as SSL/TLS in this paper. The aim of this protocol is to allow the two parties to authenticate each other and establish a shared key for subsequent communication. To analyze such protocols with respect to the various possible attacks, it is necessary to provide a brief description of the aims, assumptions, and methods of the attacks that are considered in this paper.

1.1 Public-Key-based Authentication Protocols

Authentication is usually required when two entities, A and B, that may not know each other prior to the communication, must establish each other's identity. Public key cryptography has made such authentication processes more efficient and secure than the symmetric methods previously used. During the early days of public-key cryptography [4], a rather simplistic approach to its use in providing integrity and security for digital messages was proposed: a yellow-page directory with name and public key pairs was to be distributed just like a telephone book. Soon after, to ensure that names and public keys were securely bound, certificates [9] were introduced. A certificate $\{A, K_A^+\}_{K_{CA}^-}$ binds the name A of a principal with its public key K_A^+ through a signature using the signature key K_{CA}^- of a trusted third party CA, also known as the Certificate Authority. The secrecy of a message M encrypted by this public key, $\{M\}_{K_A^+}$ is guaranteed by the fact that only the principal with the corresponding private key K_A^- can decrypt this message. On the other hand, the fact that A can decrypt a message encrypted with K_A^+ proves that A possesses the corresponding secret key, hence proving A's identity.

Certificates alone are not enough to provide the required security for an environment such as the Internet. A set of protocols along with methods to guarantee that certificates are fresh is required. The concept of a PKI was later envisioned so that the validity of all certificates can be verified by the possession of the root verification key. To solve the problem that valid certificates can become invalid over time for various reasons, a CRL which provides certificate freshness proof is also provided in a PKI.

1.2 Assumptions about the Protocol Environment

In theory, any principal who believes that his or her certificate is compromised can revoke that certificate by putting it on the CRL.

† Information Media Center, Science University of Tokyo
†† Department of Information Systems, Science University of Tokyo
††† Department of Industrial Administration, Science University of Tokyo

Principals intending to use a certain certificate can check its validity by searching the CRL. More recently, to solve the problem of the client being required to comb through all the CRLs before knowing if a certificate is invalid, the OCSP was proposed. As of today, there does not seem to be any solution that can provide an absolute guarantee of a certificate's validity and freshness at all times. On the other hand, commercial products such as the popular Netscape browser have incorporated SSL/TLS. The security of certificate-based protocols in less than ideal situations needs to be analyzed.

The SSL/TLS protocol has been subjected to considerable scrutiny over the years. Although the correctness of the protocol has been attested, various attacks [11),13),16),19)] have also been reported. These attacks have mostly involved the security of the crypto system, version rollback, and so on. Attacks using compromised certificates with inadequate CRL might seem very trivial at first glance. It is shown in this paper, however, that compromised certificates are something we have to live with and perhaps, with some extra effort, something that we can live with.

The rest of the paper is organized as follows: In Section 2 the SSL/TLS handshake protocol is described in detail. In Section 3, the scope and assumptions of our analysis are presented along with a detailed description of the "ex-employee" attack, and a proposed fix to prevent such an attack. This is followed by explanations of the logic-based and model-checking-based analysis methods in Sections 5 and 6, respectively. In Section 7, we conclude by suggesting that attacks with compromised certificates are a real threat and that effective methods should be used to prevent them.

2. SSL/TLS Handshake Protocol

In this section, we first give a brief description of the aims and requirements of public-key-based authentication protocols. This is followed by a detailed description of the "named-server, anonymous-client" version of the SSL/TLS handshake protocol.

2.1 Aim and Requirements of the SSL/TLS Protocol

The SSL/TLS protocol runs above TCP/IP and below application layer protocols such as HTTP, and LDAP, etc. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL/TLS-enabled server to authenticate itself to an SSL/TLS-enabled client. In addition, the client can authenticate itself to the server, and allows both parties to establish an encrypted connection. The SSL/TLS protocol is divided into two stages: the handshake protocol, whose aim is to establish authentication and to generate an master secret to be used in the second stage where all communication are encrypted.

The aim of the SSL/TLS protocol is to achieve certain combinations of the following three subgoals: server authentication, client authentication, and encrypted connection. The most often used model of the SSL/TLS protocol today is that of server authentication and an encrypted channel. For example, when we use on-line banking or on-line shopping Web sites, we are concerned with the identity of the Web server we are connecting to. Furthermore, communicated information such as credit card and account numbers must be encrypted. From the online bank or shop's point of view, you are authenticated by your pin number or credit card number. Although it is possible to use SSL/TLS as an anonymous encrypted channel, or an authenticated session without encryption, they are not used in business-related transactions in which both integrity and secrecy are required.

2.2 SSL/TLS Handshake Protocol Specification

In the following, we give a brief description of the SSL/TLS handshake protocol by listing the messages communicated between the client and the server. The most frequently used mode of the SSL/TLS protocol is the "named-server anonymous-client" mode, in which only the server, such as an Internet shopping mall, is authenticated. This protocol generates a master secret between an anonymous client and a named server. **Figure 1** shows the notations used in this paper.

Following Dierks and Allen [3)] and Eaves [5)], the six messages of the protocol are shown in **Fig. 2**.

In addition,

$$K_{CS} = F(N_C, N_S, N'_C) \tag{1}$$

and

CS_5 = "server finished,"
CS_6 = "client finished."

The messages can be summarized as follows:

M_1: C sends a timestamp and a nonce to S;
M_2: S sends a different timestamp and nonce to C;

C	client
S	server
CA	certificate authority
T_i	timestamp generated by i
N_i	nonce generated by i
K_i^+	public key of i
K_i^-	private key of i
$\{i, K_i^+\}K_{CA}^-$	i's public key certificate
$\{\ldots\}_{K_i^-}$	signed with K_i^-
$\{\ldots\}_{K_i^+}$	encrypted with K_i^+

Fig. 1 Notations used to describe the handshake protocol.

$$
\begin{array}{lll}
C \rightarrow S: & (N_C, T_C) & (M_1) \\
S \rightarrow C: & (N_S, T_S) & (M_2) \\
S \rightarrow C: & \{S, K_S^+\}_{K_{CA}^-} & (M_3) \\
C \rightarrow S: & \{N'_C\}_{K_S^+} & (M_4) \\
S \rightarrow C: & \{H(K_{CS}, CS_5, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_5) \\
C \rightarrow S: & \{H(K_{CS}, CS_6, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_6)
\end{array}
$$

Fig. 2 List of messages in the "named-server, anonymous-client" version of the SSL/TLS handshake protocol.

M_3: S sends its certificate to C;
M_4: C returns the "master secret" N'_C encrypted under K_S^+;
M_5: S sends a hash of the session key, a tag CS_5 indicating the protocol stage, and all preceding messages sent by S to C;
M_6: C sends a hash of the session key, a tag CS_6 indicating the protocol stage, and all preceding messages sent by C to S.

The master key K_{CS} is also used by the record layer to encrypt all communications from this point on.

2.3 Problem Statement

According to the SSL/TLS documentation [3),7)], the SSL/TLS client and server rely on information contained in the certificate to derive the necessary authentication information:

- the user certificate validity period is checked to see if today's date is within the validity period; if so,
- the user certificate issuer is checked against a list of trusted CA names; if it is in the list,
- the trusted CA certificate is used to validate the signature on the user certificate; if the signature is valid,
- the domain name in the user certificate is matched against the domain name of the principal; if the names match,
- the principal is authenticated.

A "no" answer in any of the above steps leads to immediate aborting of the handshake session. In the client authentication mode, the server optionally checks a CRL to see whether the client certificate has been revoked or not. The client, however, is not required to check whether the server certificate is fresh. The use of CRL and OCSP is outside of the specification of the SSL/TLS protocol.

In the following, we further state the following assumptions:

Certificate validity assumption:

- It is impossible for a third party to fake such a certificate;
- Certificates have unique names, and it is impossible for a client to mistake a third party's certificate for that of the server;
- The signature key of the CA is secure;
- The server optionally checks the client certificate against a CRL during each session;
- The client does not have the obligation to check the server certificate [7)].

We will also make the following assumptions about the compromised certificates.

The inadequacy of CRL has been pointed out by various researchers [1),6)] and similar attacks on password protocols using compromised certificates have been described in Halevi and Krawczyk [8)].

Compromised certificate assumption:

- An intruder has access to the private key of the server's compromised certificate;
- The owner of the certificate immediately discovers the compromise;
- The owner revokes the certificate;
- The owner obtains a new certificate from the CA.

The above assumption describes a plausible scenario in today's Internet-based use of certificates such as those used by the Netscape browser and Microsoft's Internet Explorer. For example, if a webmaster for an online bank is fired, the bank will assume that the corresponding certificate is compromised. If the bank is sensible enough, it will revoke the certificate and obtain a new certificate for future use.

We will first informally argue that under the above assumption, the SSL/TLS protocol is secure. Let us assume that the intruder I replaces message M_3 in the SSL/TLS protocol, described in Section 2.2, with the compromised

certificate. Upon receiving the compromised certificate, C successfully verifies it using the root verification key, but without checking with the CRL. When receiving M_4, S will find out that it is unable to decrypt it, since S now has a new secret key corresponding to the new certificate. The protocol run will be aborted and C will then obtain the new certificate. It may be argued that since N'_C is just a random number, S will simply decrypt M_4 without noticing that it was encrypted with a different key. In that case, the calculated master secret K_{CS} will be different for C and S, and the protocol will be aborted after message M_5. The argument goes as follows: "since the server S now has the secret key corresponding to the new, uncompromised certificate, the use of the compromised certificate will be discovered by the server and the client will update its database of certificate." Unfortunately, as shown in next section, the above arguments are false.

3. Attack on the SSL/TLS Handshake Protocol

In previous section, we gave a concrete description of the "named-server anonymous-client" version of the SSL/TLS handshake protocol. This is followed by a problem statement giving the assumptions made about the environment and the problem we are addressing. In this section, the detailed attack trace is presented with discussion of its implications.

3.1 How to Steal the Master Secret

In the following, we describe the steps showing how an intruder with the private key of a compromised certificate can learn the master secret even if the server has updated its certificate. A message in the form "$\ldots \rightarrow (X)I$" indicates that a message intended for X is intercepted by I. A message in the form "$(X)I \rightarrow \ldots$" indicates a message faked by I as from X. Note that $\{S, K_S'^{+}\}_{K_{CA}^{-}}$ is the compromised certificate of S, while $\{S, K_S^{+}\}_{K_{CA}^{-}}$ is the fresh and valid certificate of S. M'_i and M''_i are messages that are intercepted and faked by the intruder, respectively. They are in other respects identical in format with M_i. This is shown in **Fig. 3**.

In the following, we briefly explain the meanings of some of the above messages.

- M_1 and M_2 are sent in plain text;
- In M'_3 the valid certificate for S is intercepted;
- The intruder replaces it with the compromised certificate;
- In M'_4, the intruder intercepts the message and learns N'_C, the pre-master secret;
- The intruder then fakes M'_4, using the public key contained in the valid certificate;
- Since the intruder knows N_C, T_C, N_S, T_S, and N'_C, he or she is able to calculate the master secret K_{CS};
- The interception and faking of the last four messages are now possible and required, since both C and S must maintain a consistent record of the past messages despite the different versions of M_3 and M_4 kept by C and S, respectively.

$C \rightarrow S:$	(N_C, T_C)	(M_1)
$S \rightarrow C:$	(N_S, T_S)	(M_2)
$S \rightarrow (C)I:$	$\{S, K_S^{+}\}_{K_{CA}^{-}}$	(M'_3)
$I(S) \rightarrow C:$	$\{S, K_S'^{+}\}_{K_{CA}^{-}}$	(M''_3)
$C \rightarrow (S)I:$	$\{N'_C\}_{K_S'^{+}}$	(M'_4)
$(C)I \rightarrow S:$	$\{N'_C\}_{K_S^{+}}$	(M''_4)
$S \rightarrow (C)I:$	$\{H(K_{CS}, CS_5, (M_1, M_2, M'_3, M''_4))\}_{K_{CS}}$	(M'_5)
$(S)I \rightarrow C:$	$\{H(K_{CS}, CS_5, (M_1, M_2, M''_3, M'_4))\}_{K_{CS}}$	(M''_5)
$C \rightarrow (S)I:$	$\{H(K_{CS}, CS_6, (M_1, M_2, M''_3, M'_4))\}_{K_{CS}}$	(M'_6)
$(C)I \rightarrow S:$	$\{H(K_{CS}, CS_6, (M_1, M_2, M'_3, M''_4))\}_{K_{CS}}$	(M''_6)

Fig. 3 List of messages describing the "ex-employee" attack on SSL/TLS handshake protocol.

Note that M_5 and M_6 were designed to prevent attacks based on interception and faking of messages. For example, if M'_5 is allowed to reach C, C will find out that the hash of all messages does not match, because C expects (M_1, M_2, M''_3, M'_4) but instead receives (M_1, M_2, M'_3, M''_4). We can see from the above that it is ineffective in preventing the interception and faking described above, because the master secret K_{CS} is known to the intruder. After the verification steps, an SSL/TLS handshake is completed between C and S. Unfortunately, the master secret is now known by the intruder I.

3.2 Discussions

At first glance, the attack described above might be dismissed as a trivial man-in-the-middle attack. However, there are fundamental differences between the two.

3.2.1 Man-in-the-Middle Attack

Two man-in-the-middle attacks were envi-

sioned by the SSL/TLS designers [3),7)]. The first one assumes an attacker who intercepts all communication between the client and the server. The attacker intercepts the legitimate certificate and keys that are passed back and forth during the handshake, substitutes his or her own, and makes it appear to the client that he or she is the server, and to the server that he or she is the client. This type of attack is easily prevented if the domain name contained in the certificate is carefully checked. However, such procedures cannot prevent the "ex-employee" attack. The second man-in-the-middle attack described by the SSL/TLS designer concerns an attacker who is not interested in the contents of the communication, but intends to damage the communication by garbling the encrypted messages. This type of attack is prevented by attaching a message authentication code along with the message before encryption. However, this solution is also irrelevant to the "ex-employee" who is interested in stealing the master secret.

3.2.2 Other Modes of SSL/TLS Handshake Protocol

The "ex-employee" attack, however, is ineffective against the "named-sever, named-client" version of the SSL/TLS handshake protocol. In the "named-sever, named-client" version of the SSL/TLS protocol, both clients and server are authenticated. In message M_4 of this version of the protocol, C's certificate and a signed hash of the list of previous messages so far transmitted, in addition to the pre-master secret encrypted with S's public key, are sent by C to S. M_4 is shown as follows:

$$C \rightarrow S : \{C, K_C^+\}_{K_{CA}^-}, \{N'_C\}_{K'^+_S},$$

$$\{\ldots, \mathrm{Hash}(M_1, M_2, M''_3), \ldots\}_{K_C^-}(M_4)$$

The parts shown as "..." are tags and other parameters such as the master secret computed by C at this time, and are not important for the following discussion. Again, the intruder I can learn the pre-master secret N'_C by using K'^-_S, and compute the master secret K_{CS}, but is unable to fake the signed portion of M_4 because he or she does not have the signature key K_C^- of the client C. In order to maintain the consistency of the list of transmitted messages on both C and S sides, the signed part, as well as the encrypted pre-master secret in M_4, must also be altered. Since the intruder is unable to alter the signed portion of M_4, it is possible now to detect the interception and faking of M_3 and M_4 in step M_5 or M_6. The protocol run will be aborted and the attack will fail.

$$\begin{array}{lll} C \rightarrow S : & (N_C, T_C, K_C^+) & (M_1) \\ S \rightarrow C : & (\{N_S\}_{K_C^+}, T_S) & (M_2) \\ \ldots & & \end{array}$$

Fig. 4 Improved SSL/TLS handshake protocol: Method 1, basic version 1.

3.3 An Improved SSL/TLS Handshake Protocol

The "ex-employee" attack against the "named-server anonymous-client" version of the SSL/TLS handshake protocol succeeds because the intruder learns the pre-master secret N'_C, along with the other two nonces N_C and N_S transmitted in plain text. He or she can calculate the master secret K_{CS} and succeed in faking the `client_done` and `server_done` messages. It would seem that, since (1) the client is anonymous and therefore cannot generate any authenticated messages and (2) the server certificate is compromised, it is impossible to prevent this attack without requiring the client to have its own certificate.

In the following, we propose an improved handshake protocol that helps to prevent the "ex-employee" eavesdropping attack described in the previous section. It relies on a public key pair generated by the client for each session.

3.3.1 Improved Handshake Protocol: Basic Version 1

The reader is referred to Figs. 1 and 2 of Section 2 for the notations used below. To provide a certain unique identity for the "anonymous" client, a public key pair is generated by the client for each session. The public key K_C^+ of this key pair will be used by the server to encrypt the nonce N_S in the server's hello message M_2. Hence the basic version 1 of the protocol will be as shown in **Fig. 4**.

From Eq. (1) in Section 2, the master session key K_{CS} is derived from N_C, N_S, and N'_C. Even if the intruder learns N'_C because he or she has control over a compromised server certificate, he or she is unable to learn N_S encrypted with the client public key, since he or she does not have the private key generated by the client for this particular session. However, the intruder can generate a public key pair K'^+_C, K'^-_C and replace the client public key with his or her own:

$$
\begin{array}{lll}
C \rightarrow I(S): & (N_C, T_C, K_C^+) & (M_1) \\
I(C) \rightarrow S: & (N_C, T_C, K_C'^+) & (M_1') \\
S \rightarrow I(C): & (\{N_S\}_{K_C'^+}, T_S) & (M_2) \\
I(S) \rightarrow C: & (\{N_S\}_{K_C^+}, T_S) & (M_2')
\end{array}
$$

3.3.2 Improved Handshake Protocol: Basic Version 2

To prevent this simple attack, we break message M_2 into two parts. The first part will consist of the hash of the encrypted server nonce $H(\{N_S\}_{K_C^+})$. The second part of the message, the encrypted sever nonce $\{N_S\}_{K_C^+}$, will be sent after message M_4. The handshake protocol now looks as shown in **Fig. 5**.

Obviously the intruder can still replace the client public key with his or her own in message M_1, but he or she is unable to know N_S at step M_2 because he or she only has the hash of server nonce N_S. Before the client is to send message M_3, the intruder must either (1) pass the encrypted hash to the client as is, or (2) invent a completely new server nonce N_S'. In case (1), the intruder must also pass $M_{4.1}$ as is, in which case he or she will not be able to find out N_S; therefore, he or she cannot learn K_{CS}. In case (2), the intruder is forced to invent a public key pair and invent a server nonce N_S', since the client expects the encrypted hash of the sever nonce. As a result, the client and server will have different server nonces N_S' and N_S. From equation (1) we see that the master session key K_{CS} calculated by the server, and K_{CS}' calculated by the client, will be different. The client and server cannot agree on M_5 and M_6 and the protocol will be aborted.

This leaves the intruder only the option of mounting an active man-in-the-middle attack. In this case, the intruder will also fake $M_{4.1}$ by replacing N_S with N_S'. He or she will also need to generate M_5 and M_6 using K_{CS} and K_{CS}', and send them to the client and server, respectively. This way, the server and client will be able to complete the handshake run, but with a different master session for the subsequent communication. The intruder is forced to play the man-in-the-middle in this case. Otherwise, the protocol will be stopped, because the messages will not make sense after decryption. The intruder is therefore required to intercept each message from the server and decrypt them with K_{CS} and re-encrypt them with K_{CS}'. He or she also needs to intercept each message from the client and decrypt them with K_{CS}' and then re-encrypt them with K_{CS}. This is much more difficult than the passive eavesdropping attack described in Section 3, in which the intruder can decrypt all consequent communications after learning K_{CS}. All he or she needs to do is to record the encrypted communication and decrypt them off-line at a later time.

$$
\begin{array}{lll}
C \rightarrow S: & (N_C, T_C, K_C^+) & (M_1) \\
S \rightarrow C: & (H(\{N_S\}_{K_C^+}), T_S) & (M_2) \\
S \rightarrow C: & \{S, K_S^+\}_{K_{CA}^-} & (M_3) \\
C \rightarrow S: & \{N_C'\}_{K_S^+} & (M_4) \\
S \rightarrow C: & \{N_S\}_{K_C^+} & (M_{4.1}) \\
S \rightarrow C: & \{H(K_{CS}, CS_5, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_5) \\
C \rightarrow S: & \{H(K_{CS}, CS_6, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_6)
\end{array}
$$

Fig. 5 Improved handshake protocol: Basic version 2.

3.3.3 Improved Handshake Protocol: Final Version

To minimize the changes that need to be made to the current SSL/TLS handshake protocol, we can integrate the above approach by adopting the following tactics:

- **Tactic 1:** Let K_C^+ be N_C. Instead of generating a random nonce, the client is required to generate a public key pair and send the public key as N_C;
- **Tactic 2:** Let $H(\{N_S'\}_{K_C^+})$ be N_S. In addition to generating a random nonce N_S', the server is required to encrypt the hash of the random nonce with the K_C^+ he or she received from the client in M_1, and send it as N_S;

The final version of the improved handshake protocol is shown in **Fig. 6**. Notice that the N_C and N_S are now generated in accordance with Tactics 1 and 2.

$$
\begin{array}{lll}
C \rightarrow S: & (N_C, T_C) & (M_1) \\
S \rightarrow C: & (N_S, T_S) & (M_2) \\
S \rightarrow C: & \{S, K_S^+\}_{K_{CA}^-} & (M_3) \\
C \rightarrow S: & \{N_C'\}_{K_S^+} & (M_4) \\
S \rightarrow C: & \{N_S'\}_{K_C^+} & (M_{4.1}) \\
S \rightarrow C: & \{H(K_{CS}, CS_5, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_5) \\
C \rightarrow S: & \{H(K_{CS}, CS_6, (M_1, & \\
 & M_2, M_3, M_4))\}_{K_{CS}} & (M_6)
\end{array}
$$

Fig. 6 Improved handshake protocol: Final version.

N'_S is the original nonce used to generate N_S using Tactic 2, and

$K_{CS} = F(N_C, N'_S, N'_C)$.

Notice that the client only learns N'_S after message $M_{4.1}$ instead of after message M_2 as in the original protocol.

4. How the Above Attacks Were Discovered

We have been interested in using both the model checking method [17] and the logic-checking method [15] for analyzing authentication protocols. First, we extended the BAN logic [2] to allow it to deal with the analysis of secrecy theorems, and succeeded in discovering a weakness in the fix [10] proposed by Gavin Lowe for the Needham-Schroeder public key authentication protocol. Model checking is then used to discover the exact attack for the described weakness. It turned out that the weakness discovered in the Needham-Schroeder protocol affects the "named-server anonymous-client" mode of SSL/TLS handshake protocol.

The next two sections summarize the results of our analysis of the Needham-Schroeder authentication protocol using both methods. Readers interested in the details should consult two of our previous papers [15),18].

5. Extending BAN Logic for Secrecy Analysis

BAN logic was proposed for analyzing the properties of authentication protocols such as Kerberos, Needham-Schroeder, and SSL, and led to the discovery of a number of weaknesses in various protocols. However, due to its modeling of the principals as benign agents, as pointed out by Dan Nessett [12], it is considered ineffective for analyzing secrecy-related properties of authentication protocols. As described in Nessett's paper [12], the environment modeled in BAN does not include principals that are not supposed to gain access to certain secrets. Since such principals are not modeled, it is impossible to analyze properties related to secrecy. Nessett further illustrated this weakness by describing an obviously flawed protocol, and proved the possibility of proper authentication using the BAN logic.

The criticism is somewhat unfair in that the designers of BAN logic have never claimed it can be used to deal with secrecy. Only proper authentications, as defined by a set of beliefs, were defined and used in the analysis.

Our work is partially motivated by this criticism, as well as other recent research in dealing with the analysis of secrecy properties of authentication protocols. For example, Paulson has incorporated a secrecy theorem into his logic for security protocols [13]. Roscoe described a model for a "spy" [14]. Lowe successfully discovered an attack [10] on the Needham-Schroeder public key authentication protocol by using a model of the protocol including an intruder. Other results on verification of SSL [11] and TLS [5),13] also encouraged our investigation.

5.1 Parameterizing of BAN Logic

In the following, the Needham-Schroeder public key authentication protocol is described briefly before the parameterizing is explained using this protocol as an example. From now on, we will refer this protocol as the original protocol. In the simplified version of this protocol, only two principals, an initiator A and a responder B, are considered. Three messages are exchanged:

$$\begin{aligned} A \rightarrow B: &\quad \{N_A, A\}_{K_B^+} &\quad (M_1)\\ B \rightarrow A: &\quad \{N_A, N_B\}_{K_A^+} &\quad (M_2)\\ A \rightarrow B: &\quad \{N_B\}_{K_B^+} &\quad (M_3) \end{aligned}$$

We introduce a third principal, an intruder I and model the above protocol using parameterizing. First we fix the initiator A, then we parameterize the responder by a variable X_B, where $X_B \in \{B, I\}$, This allows us to model a protocol run that includes possible interception and faking of messages by the intruder. For example, the following message:

$$A \rightarrow X_B : \{N_A, A\}_{K_{Y_B}^+}, \text{where } Y_B \in \{B, I\},$$

can be interpreted as

- $X_B = Y_B = B$: a message sent from A to B encrypted with B's public key, which corresponds to the normal message described in the original protocol definition;
- $X_B = Y_B = I$: a message sent from A to I using I's public key;
- $X_B = B, Y_B = I$: a message sent from A, intended for B, using I's public key.

Note that we use Y_B and X_B to indicate that the intended receiver and its public key bearer may be different.

5.2 Secrecy Property in Parametrized BAN

With the introduction of an intruder I, we are in the position to define the secrecy property for the above protocol.

Definition 1 If both shared secrets N_A and N_B, intended for an authentication session between A and B, can be obtained by intruder I, the secrecy property of the protocol is considered to be violated. In terms of the parametrized BAN logic, if the formulas corresponding to "I saw N_A" and "I saw N_B" can be derived from the protocol, we regard the secrecy of the protocol as violated.

For the original Needham-Schroeder protocol described above, both formulas corresponding to "I saw N_A" and "I saw N_B" can be derived [15]. This indicates that the secrecy property for the protocol is violated. Unfortunately, the analysis does not directly point to an attack. Lowe [10] discovered the following attack by using the model-checking method. We will refer to this attack as the "impersonation attack."

$$\begin{array}{lll} \alpha: & A \rightarrow I & \{N_A, A\}_{K_I^+} \\ \beta: & (A)I \rightarrow B & \{N_A, A\}_{K_B^+} \\ \beta: & B \rightarrow (A)I & \{N_A, N_B\}_{K_A^+} \\ \alpha: & I \rightarrow A & \{N_A, N_B\}_{K_A^+} \\ \alpha: & A \rightarrow I & \{N_B\}_{K_I^+} \\ \beta: & (A)I \rightarrow B & \{N_B\}_{K_B^+} \end{array}$$

The above trace describes two interleaving runs α and β. Run α is between A and I, and run β is between $(A)I$ and B. $(A)I$ indicates that an intruder I impersonates A. Lowe proposed a fix [10] to deny this attack. In the fixed protocol, Message 2 is

$$B \rightarrow A : \{N_A, N_B, B\}_{K_A^+}$$

We will refer to this addition as "Lowe's fix". This prevents the "impersonation attack," since Message 2 cannot be simply copied from run β to run α. A will expect I rather than B as its correspondent. This fix, however, is shown to be ineffective against the "ex-employee attack" described in Section 3.

5.3 Security of the Fixed Needham-Schroeder Protocol

We idealize Message 2: $\{N_A, N_B, B\}_{K_A^+}$ in the fixed Needham-Schroeder protocol by adding the substitution term $X_B = B$ to our model of the Needham-Schroeder protocol using parametrized BAN logic. In this case, the inference results do not contain the formula "I see N_B." Formulas corresponding to the proper authentication described in the original BAN logic can also be obtained.

This result appears to indicate that the fixed Needham-Schroeder is secure. However, on closer examination, the secrecy property holds if the assumption $X_B = B$ is guaranteed. The security of the fixed protocol is therefore dependent on this assumption. This turned out to be the weakness of the fixed Needham-Schroeder protocol.

Quick discussion: Analysis based on logic, such as the BAN logic, does not directly point to possible attacks. Rather, it highlights the weaknesses in the protocol by pointing to certain weak assumptions that may not stand up to close scrutiny. Further analysis of such assumptions may lead to the discovery of an actual attack. Our study also follows this pattern. In the next section, we show how an attack on this fixed protocol can be discovered.

6. Discovery of Actual Attacks by Means of Model Checking

Unlike logic-based methods such as the BAN logic, in which messages are abstracted and idealized as logical formulas representing certain beliefs held by each principal, the model-checking method requires a concrete state-based model to be constructed. Usually, each principal is represented by a process. States of the process, such as "running" or "committed," are reached depending on the messages that are sent or received. The model checker provides a trace of an actual attack if an assertion or a temporal logic formula describing a certain requirement specification is violated.

In the following, we describe a state-based model of the protocol that includes an initiator, a responder, and an intruder. The secrecy requirements used to discover the "ex-employee attack" are also described.

6.1 State-based Model of Needham-Schroeder Protocol

The state-based model consists of three concurrent processes representing the initiator, the responder, and the intruder. The initiator and the responder send and receive messages strictly according to the protocol specification. The intruder, however, is allowed to perform the following actions:

- overhear and store a copy of the message;
- steal/intercept a message from its intended receiver;
- decrypt messages that are encrypted with its public key;
- replay any stored message at any time;

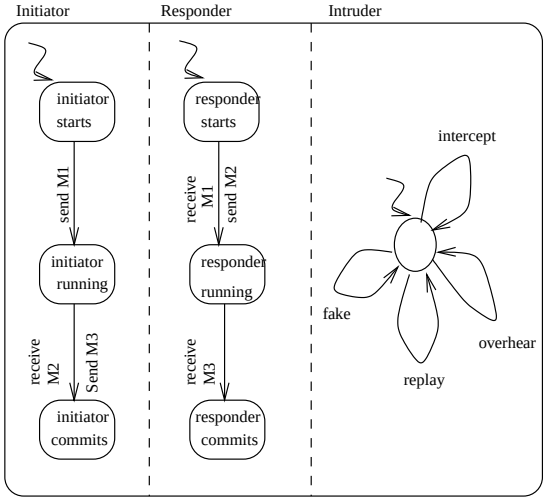


Fig. 7 The state-based model for Needham-Schroeder protocol.

- make up new messages using learned secrets such as stolen nonces.

The above assumptions are based on Lowe [10] and justified in the current Internet environment, assuming that the cryptography used by the protocol is strong enough. An illustration of the model is shown in **Fig. 7**.

From the results described in Section 5.3, it is clear that the security of Lowe's fix depends on whether we can guarantee $X_B = B$ when receiving Message 2. This is assumed in Lowe's analysis. To further explore the consequences of deliberately weakening this assumption, we allow the possibility that A may not have the means of positively identifying X_B as B. In other words, it is possible for A to send a message intended for B by using a public key that may or may not be B's. This is to say that messages of the format

$$A \rightarrow B : \{N_A, A\}_{K_I^+}$$

are not excluded from our model.

This is the only difference between our protocol model and Lowe's.

6.2 Nonce Secrecy Requirements

In Lowe [10], the requirements for proper authentication are described as safety properties:

- If the responder is committed to the initiator, the initiator must be running with the responder;
- If the initiator is committed to the responder, the responder must be running with the initiator.

It is quite clear that such requirements are concerned with the possibility of the intruder impersonating either the initiator or the responder.

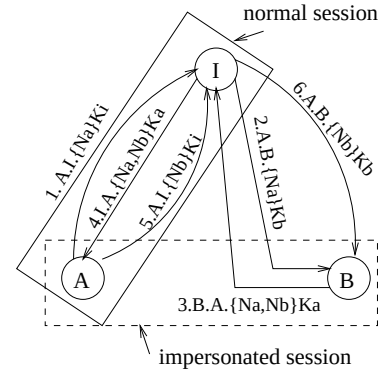


Fig. 8 The "impersonation attack" due to Lowe.

Our analysis results given in the logic-checking phase, however, point to the necessity of protecting the secrecy of the nonces. Consequently, our requirements are given as a safety property of the secrecy of the nonces N_A and N_B.

- Intruder I must not be able to learn both N_A and N_B in protocols between A and B.

6.3 Results of Model Checking

Two versions of the Needham-Schroeder protocol, the original Needham-Schroeder and the version with Lowe's fix were model checked using the secrecy requirement described in the previous subsection.

Both the "impersonation attack" and the "ex-employee" were found in the results of model-checking the original version of the protocol. The results are shown in **Figs. 8** and **9**, respectively. Note that the first attack was discovered by using the secrecy requirements rather than the proper authentication requirement used by Lowe. In fact, Lowe's attack can also be described as the result of I learning N_A and N_B, which is supposed to be a shared secret between A and B.

The model-checking result for the version with Lowe's fix only produced the attack corresponding to the "ex-employee." In other words, Lowe's fix is effective against the "impersonation attack" but not against the "ex-employee attack." The readers can verify this by modifying Message 3 in Fig. 9 and see that it was not effective in preventing the "ex-employee attack."

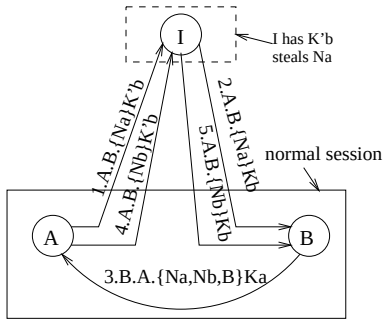


Fig. 9 The "ex-employee attack" due to Wen-Saito.

6.4 Another Fix for Needham-Schroeder

The "ex-employee" attack described in the parameterized BAN analysis assumes that the public key of A is secure, as in the case of model-checking. This need not be the case in reality. It is possible that A's public key might be compromised and that the same attack might succeed. However, the likelihood of both A and B's certificates being compromised for the same attacker is negligible, so we will not be concerned with this case.

We propose a fix that can eliminate both the "impersonation attack" and the "ex-employee attack" on the Needham-Schroeder protocol. Instead of adding only B to Message 2 in the original protocol, we propose to replace A with K_A^+ in Message 1, and to replace B with K_B^+ in Message 2. The fixed protocol is as follows:

$$\begin{aligned} A \to B: &\quad \{N_A, K_A^+\}_{K_B^+} && (M_1)\\ B \to A: &\quad \{N_A, N_B, K_B^+\}_{K_A^+} && (M_2)\\ A \to B: &\quad \{N_B\}_{K_B^+} && (M_3) \end{aligned}$$

Since the intruder I is unable to decrypt Message 2 and replace K_B^+ with $K_B'^+$, A will be able to detect the inconsistency. The same argument can be used for the situation when A's certificate is compromised. Even if the intruder chooses not to check for inconsistency of the public keys contained in the messages, the subsequent message will be encrypted with the new key contained in the message, and the intruder will be unable to learn both nonces, which is necessary in order to decrypt the subsequent communication.

7. Conclusions

When analyzing public-key-certificate based protocols, it is normally assumed that the validity and freshness of the certificate is guaranteed by the deployed PKI. In reality, methods for guaranteeing such freshness, such as CRL and OCSP, are either ineffective or difficult to incorporate into the various authentication protocols used currently. The case of the total loss of security as the result of a compromised certificate, where the owner is not aware of the problem, does not seem to have a solution. If, however, one of the principals engaged in the authentication run has a compromised certificate, knows about it, and replaces it with a new valid certificate, it appears that the protocol should be safe, since the owner will detect such inconsistency and stop the run. The results described in this paper show that an attack taking advantage of a "known" invalid certificate exists for the "named-sever anonymous-client" version of the most popular authentication protocols used today. A fix that requires some modification of the current SSL/TLS handshake protocol has been proposed. We believe that there is a valid reason for making public-key-certificate based authentication protocols resistant to compromised-certificate-based attacks such as the "ex-employee" attack described in this paper.

Acknowledgments The autors would like to thank Frank Scheneider at JPL for helpful comments and discussions.

References

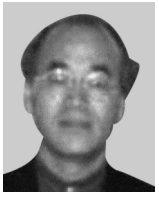
1) Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A.: The Role of Trust Management in Distributed System Security, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, Springer-Verlag (1999).
2) Burrows, M., Abadi, M. and Needham, R.: A Logic of Authentication, Technical Report 39, DEC Systems Research Center (Feb. 1989).
3) Dierks, T. and Allen, C.: The TLS Protocol: Version 1.0. Technical Report drat-ietf-tls-rptocol-05.txt.Z, IETF task force (May 1998).
4) Diffie, W. and Hellman, M.: New Directions in Cryptography, *IEEE Trans. Inf. Theory*, Vol.22, No.6, pp.644–654 (1976).
5) Eaves, W.D.: Transport Level Security: A Proof Using the GNY Logic, Technical Report, Brunel University, UK (Feb. 1989).
6) Elliosn, C., et al.: SPKI Certificate Theory, Technical Report, IETF Task Force (Nov. 1997).
7) Freier, A., Kocher, P. and Kaltorn, P.: SSL V3.0 Specification, Technical Report, http://home.netscape.com/eng/ssl3/s-SPEC.HTM, IETF Task Force (Mar. 1996).
8) Halevi, S. and Krawczyk, H.: Public-Key

Cryptography and Password Protocols, *ACM Trans. Information and System Security*, Vol.2, No.3, pp.230–268 (1999).
9) Kohnfelder, L.: Towards a Practical Public-Key Cryptosystem, Technical Report, MIT (1978).
10) Lowe, G.: Breaking and Fixing the Needham-Schroeder Public Key Protocol Using CSP and FDR, *TACAS96* (1996).
11) Mitchell, J., Mitchell, M. and Stern, U.: Automated Analysis of Cryptographic Protocols Using Murphi, *IEEE Symposium on Security and Privacy*, pp.141–151 (1997).
12) Nessett, D.: A Critique of the Burrows, Abadi and Needham Logic, *ACM Operating Systems Review*, Vol.24, No.2, pp.35–38 (1990).
13) Paulson, L.: The Inductive Approach to Verifying Cryptographic Protocols, *J. Computer Security*, Vol.6, pp.85–128 (1998).
14) Roscoe, A.W.: The Perfect 'Spy' for Model-Checking Cryptoprotocols, *DIMACS Workshop on Design and Formal Verification of Security Protocols*, Piscataway, NJ (Sep. 1997).
15) Saito, T., Wen, W. and Mizoguchi, F.: Analysis of Authentication Protocol by Parameterized Ban Logic, Technical Report, ISEC (July 1999).
16) Wagner, D. and Schneider, B.: Analysis of the SSL V3.0 Protocol, *Second USENIX workshop on Electronic Commerce* (Nov. 1996).
17) Wen, W. and Mizoguchi, F.: Model Checking Security Protocols, *Symposium on Cryptography and Information Security*, pp.647–652, Kobe, Japan (Jan. 1999).
18) Wen, W., Saito, T. and Mizoguchi, F.: New Results of Model Checking Needham-Schroeder Protocol, Technical Report, ISEC (Oct. 1999).
19) Wen, W., Saito, T. and Mizoguchi, F.: Security of Public-Key Based Authentication Protocol, *LNCS*, Vol.1751, pp.196–209 (2000).

(Received November 30, 1999)
(Accepted June 1, 2000)

Wu Wen was born in 1966. He received his B.S. degree from Beijing University of Aeronautics in 1985, M.S. degree from University of Wales in 1988 and D.Phil degree from University of Oxford in 1992, respectively. He has worked at NTT Communication Science Labs in Japan and NASA Software Verification Facility in USA before taking up an associate professor position at the Science University of Tokyo. His current research interests are software verification and computer security. He co-chaired the 5th IEEE Enterprise Security Workshop and is a member of ACM and IEEE Computer Society.

Takamichi Saito was born in 1971. He received his M.S. degree from the Science University of Tokyo in 1998 and is currently a research associate at the same university. He is a member of Software Science Society of Japan and Electronic Information Communication Society of Japan.



Fumio Mizoguchi was born in 1941. He received his M.S. degree from the Science University of Tokyo in 1968. Since 1987 he has been a professor at the Science University of Tokyo and is the director of Information Media Center at the same university. His interests are in artificial intelligence, cognitive science and software engineering. He is an editor of Artificial Intelligence Journal and member of Software Science Society of Japan, Cognitive Science Society of Japan, Information Processing Society of Japan and American Association of Artificial Intelligence.