Multi-Protocol Attacks and the Public Key Infrastructure*

Jim Alves-Foss
Center for Secure and Dependable Software
Department of Computer Science
University of Idaho
Moscow, ID 83844-1010
email: jimaf@cs.uidaho.edu

Abstract

The public-key infrastructure will be utilized to store and disseminate certified copies of user's public keys for use in secure transmission and digital signature verification. This paper presents a class of attacks, *multi-protocol attacks*, which can be used to break otherwise secure public-key based authentication protocols. These attacks are possible when the public-key infrastructure permits the use of a user's public key in multiple protocols. An attacker can then use either an existing protocol or a *tailored protocol* to subvert an otherwise secure protocol. Possible solutions are discussed.

Keywords: Public Key Infrastructure, Cryptography, Security, Authentication

1 Introduction

The widespread use of electronic commerce and other networked applications that require a high level of authentication will benefit from the establishment of a public-key infrastructure (PKI). With such an infrastructure in place, it will be easy for users to obtain the public keys of other participants in a trusted and secure manner. This trusted distribution of public keys would allow for the widespread use of public-key based algorithms for secure and authenticated communication across public networks.

Unfortunately there is a class of attacks, *multi-protocol* that are successful against a large class of public-key authentication protocols. In implementations of the public-key infrastructure where a user's public key can be used for more than one specific protocol, public-key authentication protocols can be broken. For example, the installation of an insecure or possibly *tailored protocol* (called chosen protocols in [KSW97]) on user *B*'s machine could result in an attacker being able to masquerade as *B* to other entities.

For the purposes of this paper, we define a public-key authentication protocol as any protocol that relies on the use of public-key signatures and/or encryption to validate the identity of protocol participants. We assume that the definition of the protocols and the generation of public-key pairs is disjoint and that a public key can be used in more than one protocol. We

* Project sponsored in part by National Security Agency under Grant Number MDA904-96-1-0108. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation hereon.

concede the security of the private part of the public key within the user's machine and the security of public-key certificates as the security of these items does not affect the attacks presented here.

This paper discusses multi-protocol attacks and how they can be realized to defeat otherwise secure public-key authentication protocols. We highlight this discussion with examples of attacks on two proposed public key-based authentication protocols. We conclude with a discussion of design measures that must be taken to prevent this class of attack.

2 Authentication

We make several assumptions in the analysis of authentication protocols in this paper. Most of these assumptions can be relaxed and will not affect the efficacy of the attacks presented. These assumptions are justified as protocol designers make several of these assumptions in the design and analysis of authentication protocols. We have used them to simplify the analysis and discussion of multi-protocol attacks.

- *Perfect encryption*. We denote message M encrypted with key K by $\{M\}_K$ (if K is A's public key we write K as $PK(A)$, if K is A's private key we write K as $PKS(A)$). *Perfect encryption* means that no user can determine M from $\{M\}_K$ without knowledge of K and that $\{M\}_K = \{M'\}_{K'}$ if and only if $M = M'$ and $K = K'$. The later can be realized if we assume the inclusion of an encoded checksum within $\{M\}_K$ such that we can validate that the decryption used the correct key K. This is a common practice in many encryption schemes.
- Other than the implicit checksum within $\{M\}_K$, we require that all other fields of the message be explicitly specified (this includes direction bits, protocol sequence numbers or protocol identification fields). This requirement avoids confusion due to hidden assumptions in the presentation of protocols.
- Some portions of the message may be included in plain text. For example we specify all protocols with the plain-text source and destination identifiers as the first two fields of the protocol.
- *Implicit connection determination*. All interactions over the network are considered connection oriented. Therefore, participants can be involved in multiple protocols simultaneously without messages from those protocols being accidentally interleaved (see the attack on the Woo-Lam protocol discussed in [AN94].)
- *Implicit termination*. If a data value is specified explicitly for a field of a protocol message (such as the nonce N), occurrence of any other value in that field will result in the termination of the protocol.
- *Protocol completion.* If a participant completes the sending/receiving of the last message of the protocol, the participant accepts the authentication of the other participant and the goals of the protocol.

3 Multi-Protocol Attacks

A multi-protocol attack is an attack against an authentication protocol that uses messages generated from a separate protocol (not just another run of the same protocol) to spoof one of the participants into successfully completing the protocol. In this section we present three examples of multi-protocol attacks against secure protocols. These examples demonstrate attacks against

Message 1. $A \rightarrow S : A.S.B$
Message 2. $S \rightarrow A : S.A.\{PK(B),B\}_{PKS(S)}$
Message 3. $A \rightarrow B : A.B.\{N_a.A\}_{PK(B)}$
Message 4. $B \rightarrow S : B.S.A$
Message 5. $S \rightarrow B : S.B.\{PK(A),A\}_{PKS(S)}$
Message 6. $B \rightarrow A : B.A.\{N_a.\ N_b\}_{PK(A)}$
Message 7. $A \rightarrow B : A.B.\{N_b\}_{PK(B)}$

Figure 1. Needham Schroeder Public-Key Protocol

protocols that use public-keys for secrecy and against protocols that use public-keys for digital signatures. Other examples of attacks can be found in [KSW97].

3.1 Example 1: Attacks Against Public-Key Secrecy

The protocol depicted in Figure 1 shows the original Needham Schroeder public-key protocol (NS) [NS78]. The purpose of this protocol is to establish and authenticate communication between *A* and *B* and to exchange copies of shared secret values (nonces). The protocol proceeds as follows:

1. *A* requests a copy of *B*'s public key from the key distribution server, *S*.
2. *S* sends a signed message to *A* (hence the subscript *PKS*(*S*)) that includes a copy of *B*'s public key. Such a message should be a signed public key certificate such as those defined in PKCS#6 [RSA93a].
3. *A* then sends a secret message to *B* indicating *A*'s identity and a nonce N_a. This message is encrypted with *B's* public key. (For efficiency purposes, modern implementations may actually send a message encrypted with a message key, and send the message key encrypted with *B*'s public key [RSA93b]).
4. *B* then requests *A*'s public key from the key distribution server.
5. The server responds with a certificate, as in step 2.
6. *B* sends a message to *A* including a new nonce N_b and *A*'s original nonce N_a. The intent of this message is to permit *A* to verify that *B* is active in this protocol.
7. Finally, *A* sends the new nonce N_b back to *B* to allow *B* to verify that *A* is active in this protocol. Both participants can now use these nonces in subsequent communication, such as the foundations of a session key.

Message 3. $A \rightarrow B : A.B.\{N_a.A\}_{PK(B)}$
Message 6. $B \rightarrow A : B.A.\{N_a.N_b.B\}_{PK(A)}$
Message 7. $A \rightarrow B : A.B.\{N_b\}_{PK(B)}$

Figure 2. Modified Needham Schroeder Public-Key Protocol

Message i. $B \rightarrow A : B.A.\{M.N_b.B\}_{PK(A\}}$
Message ii. $A \rightarrow B : A.B.\{N_b.B\}_{PKS(A)}$

Figure 3. Tailored Protocol 1

If we have a trustworthy PKI we can assume that steps 1, 2, 4 and 5 of the NS protocol can be accomplished securely. This leaves us with only steps 3, 6 and 7 to worry about. Unfortunately, messages 6 and 7 do not provide enough information to *B* to validate that *A* is active in the *current protocol run*, but only to validate that *A* is active. Lowe [Low96] demonstrated a successful attack against this protocol by interleaving messages from two concurrent runs of the protocol with one untrustworthy participant able to masquerade as *A* to *B*. Lowe then modified the relevant messages of the protocol (Figure 2) to provide a secure protocol and provided a detailed proof of the security of this protocol.

Now consider the protocol in Figure 3. This protocol may have been designed by an application programmer, or network service provide, who wanted to develop a certified message receipt mechanism, or it could be a deliberately tailored protocol to break other authentication protocol. In this protocol, user *B* sends a secret message consisting of data value *M* (this may correspond to some message being sent to a higher level protocol on *A*'s machine), a nonce for freshness and identification purposes and *B*'s name to indicate the source. *A* responds with a signed copy of the nonce, hence the *PKS*(*A*) subscript, and *B*'s name. This provides a guarantee to *B* that *A* is active and received the sent message. A quick examination of this protocol shows that there is no authentication of *B* to *A*; other security problems may also exist. However, we don't care if this protocol is secure or not, since we are only going to use it to break the modified NS protocol. The only important aspect of the protocol is that message i is exactly the same format as message 6. This includes fields for sequence numbers, protocol step identifiers, protocol identifiers, field data type indicators, checksums, etc.

As we have mentioned, Lowe provided a detailed proof of the security of the modified NS protocol depicted in Figure 2. In Figure 4 we demonstrate an attack on this protocol through the interleaving of messages from multiple protocol runs, specifically the tailored protocol of Figure 3 interleaved with the modified NS protocol. The attack proceeds as follows:

Modified NS Protocol	**Tailored Protocol 1**
Message 3. $E_A \rightarrow B : A.B.\{N_a.A\}_{PK(B)}$	
Message 6. $B \rightarrow E_A : B.A.\{N_a.N_b.B\}_{PK(A)}$	
	Message i. $E_B \rightarrow A : B.A.\{M=N_a.N_b.B\}_{PK(A\}}$
	Message ii. $A \rightarrow E_B : A.B.\{N_b.B\}_{PKS(A)}$
Message 7. $E_A \rightarrow B : A.B.\{N_b\}_{PK(B)}$	

Figure 4. Multi-Protocol Attack Against Modified NS

1. The intruder, E_A, masquerading as *A* sends message 3 to *B*. The tailored protocol requires nonce N_a and *M* to have the same format.
2. The intruder will then receive the response message 6 from *B*.
3. The intruder, masquerading as *B*, forwards this message as message i for the tailored protocol to user *A*.
4. *A* responds with message ii of the tailored protocol, which includes a publicly readable copy of the nonce N_b.
5. The intruder grabs this value and creates message 7 and sends it on to *B* who then validates the intruder's identity as *A*.

The attack succeeds because the intruder is able to force *A* to decode the secret field N_b from message 6 and to send it back to the intruder in a format that the intruder can read. Since message i of the tailored protocol is formatted precisely the same as message 6 there is no way that *A* can detect the attack.

3.2 Example 2: Attacks Against Public-Key Signatures

The protocol depicted in Figure 5 is based on the draft SSL-3 protocol. The intent of the protocol is to enable server, *B*, with a signature only RSA certificate [RSA78], to authenticate itself to *A* and to securely exchange data used to create a master encryption key for subsequent communication.. All of the messages in this protocol are sent in plain text except for messages 4, 6 and 7. The protocol proceeds as follows:

1. The client initiates communication with the server by sending a client hello message that contains a list of possible cryptographic and compression routines along with an initial random number.
2. The server responds with a server hello message that contains similar information as the client hello message except that it indicates specific cryptographic and compression routines.
3. The server then sends a copy of its certificate(s), typically X.509.v3 certificates [CCITT89].
4. In this version of the protocol, the server sends a plain text copy of a temporary RSA public key (a modulus, exponent pair) and a signed hash of that key (the hash is an MD5

Message 1. $A \rightarrow B$: *Client Hello*
Message 2. $B \rightarrow A$: *Server Hello*
Message 3. $B \rightarrow A$: *Server Certificate*
Message 4. $B \rightarrow A$: $K.\{H(K)\}_{PKS(B)}$
Message 5. $B \rightarrow A$: *Server Hello Done*
Message 6. $A \rightarrow B$: $\{M\}_K$
Message 7. $B \rightarrow A$: $\{Server\ Finished\}_{K'}$

Figure 5. Proposed SSL-like Authentication Protocol

Message i. $A \rightarrow B : A.B.\{M\}_{PK(B)}$
Message ii. $B \rightarrow A : B.A.\{H(M)\}_{PKS(B)}$

Figure 6. Tailored Protocol 2

hash [Riv92] followed by a SHA hash [NIST93]). This message occurs if the server has no certificate or has a signature only certificate; the later case is the one addressed here.

5. The server then sends a server hello done message.
6. The client sends a secret message (a pre-master key) encrypted with the server's temporary RSA public key.
7. The server responds with a server finished message that is a hash of the exchanged information encrypted with the just negotiated values.

Note that message 4 includes a plain-text copy of the server's temporary RSA public-key and a digitally signed hash of that key. The client can use the server's certificate and the signed hash to authenticate the origin of the temporary public key. In message 6 the client uses this temporary public key to secretly send some data (a pre-master key) to the server. Given the security of the RSA algorithm the client can be assured that only the owner of the temporary RSA key can read the message. Thus, an attacker who tries to replay an old message 4 from the server will not be able to read message 6 from the client.

Now consider the protocol in Figure 6. This protocol may been designed by an application programmer, or network service provide, who wanted to develop an certified message receipt mechanism, or it could be a deliberately tailored protocol to break other authentication protocol. In this protocol, user *A* sends a secret message consisting of some data value *M* to the server *B*. The server responds with a signed hash of the message *H*(*M*) to authenticate receipt of the message. Again, as with the first tailored protocol, we are not concerned about the security of this protocol. However, upon completion the client is sure that the server received the secret message.

Proposed SSL-Like Protocol	**Tailored Protocol 2**
	Message i. $A \rightarrow B : A.B.\{M=K'\}_{PK(B)}$
	Message ii. $B \rightarrow A : B.A.\{H(M=K')\}_{PKS(B)}$
Message 1. $A \rightarrow E_B$: *Client Hello*	
Message 2. $E_B \rightarrow A$: *Server Hello*	
Message 3. $E_B \rightarrow A$: *Server Certificate*	
Message 4. $E_B \rightarrow A$: $K'.\{H(K')\}_{PKS(B)}$	
Message 5. $E_B \rightarrow A$: *Server Hello Done*	
Message 6. $A \rightarrow E_B$: $\{M\}_{K'}$	
Message 7. $E_B \rightarrow A$: *Server Finished*	

Figure 7. Multi-Protocol Attack Against Proposed Protocol

It is not unreasonable to believe that similar messages might exist in secure protocols. The important concept of this protocol is that it forces the server to sign a hash of a message that it did not create.

We can take advantage of this flaw in the protocol to attack the protocol of Figure 5 as demonstrated in Figure 7. Here the intruder creates its own temporary RSA public key, *K*' and sends it to *B* who responds with a signed hash of the key *H*(*K*'). The intruder then stores that value and waits for the client *A* to initiate communication to a site masquerading as the valid server. During communication the intruder will send the hashed value of its own temporary RSA public key to the client who will authenticate it as having come from the server. The client will respond with a secret message encrypted with the intruder's key. The intruder can read the encrypted message and continue to masquerade as the valid server since the client's only authentication involves the checking of a spoofed digital signature and does not include any other form of authentication. This attack forces *B* to blindly sign a message, something no secure protocol should do.

3.3 Example 3: A Second Attack on a Public-Key Signatures

Consider the protocol depicted in Figure 8. This protocol is known as the Station-to-Station (STS) protocol and was proposed by Diffie, van Oorschot and Wiener [DvOW92] and was proven secure by Syverson and van Oorschot [SvO96]. In the discussion of the proof, Syverson and van Oorschot state

> "First, we may assume that honest principals are competent enough to not encrypt or sign messages blindly, i.e., without any understanding of the message content."

If we hold that this is true, the preceding attacks are not possible. However, here we present a second attack on the SSL-like protocol that does not violate this claim. This attack is based on an interleaving of an STS-based and SSL-like protocols; both of which can be shown to be secure, and both of which have competent principals.

We make a few minor modifications and assumptions related to the STS-based protocol. These are needed for the attack to work in this particular case, but do not detract from the security of the protocol. With the proliferation of authentication protocols there is an increased likelihood that a matching of message formats between protocols, as shown here, will occur, or that an attacker could tailor a protocol as we have done. We assume that the digital signing found in the STS protocol actually consists of a digital signing of hashed values in the same format of the SSL-like protocol. This is a reasonable assumption since most cryptographic software packages provide digital signatures only over hashes, and this form of signing is defined in many of the standards. For this attack to work we assume that the RSA modulus and exponent of the

Message i. $B \rightarrow A : B.A.R_b$
Message ii. $A \rightarrow B : A.B.R_a.\mathrm{Cert}_b.\{\{H(\mathrm{R_a},R_b)\}_{PKS(A)}\}_K$
Message iii. $B \rightarrow A : B.A.\mathrm{Cert}_a.\{\{\mathrm{H}(\mathrm{R_a},R_b)\}_{PKS(B)}\}_K$

Figure 8. STS-based Protocol

SSL-like protocol and the Diffie-Hellman (DH) public values [DH76] all have the same number of bits, (e.g., 16, 32, etc.). We also assume that neither protocol embeds any protocol identifiers or other information (except for checksums) within the signed messages, and that all signing and hashing use the same versions of the same algorithms. The STS-based protocol proceeds as follows:

1. User *B* creates a DH public value R_b and sends it to *A*.
2. User *A* creates a DH public value R_a, the DH session key *K* and a token and sends the public value, *A*'s certificate and the token to *B*. The token is the pair of public values digitally signed by *A* and then encrypted with the DH session key. The signature actually only occurs over the hash of these values, contrary to the original protocol specification.
3. User *B* reads and verifies the information from *A*. *B* then creates the DH session key *K* and a token (similar to the token *A* created) and sends *B*'s certificate and the token to *A*. Note that the order of the DH public values is reversed in our version of the protocol compared with the original

Since *A* and *B* are the only entities able to digitally sign these tokens, and we assume that they are competent in signing, the participants can successfully authenticate each other's identity. However, suppose that *A* is a malicious user that wants to masquerade as *B*. Consider the following attack:

1. *E* receives a communication request from *B* with public value R_b.
2. *E* calculates two primes *p* and *q* to generate an RSA modulus $n= pq$. This is accomplished by randomly generating DH public values until two reasonable primes are found, (the density of prime numbers and the mathematical nature of the DH values guarantees that this will not take an unreasonable number of computations). The DH public values are generated by private values x_p and x_q which result in a private DH value $x = x_p + x_q$. *E* uses this value to create $R_a= n=pq$. which is sent to *B*. *E* also uses R_b as an RSA public-key exponent and using *p* and *q* generates a corresponding private RSA exponent.
3. *B* responds with its certificate and the encrypted, signed token. *E*, knowing the session key *K* is able to obtain the signed pair $\{R_b,R_a\}PKS\{B\}$ which exactly corresponds to a signed version of *E*'s RSA public-key modulus and exponent.
4. *E* now has a message in the format of message 6 of the SSL-like protocol, signed by *B*, and can proceed the same as the previous attack.

4 Understanding and Preventing the Attacks

The examples presented in the previous section demonstrate a broad class of tailored protocols that can be used to attack existing, otherwise secure protocols. This section describes the environment under which these attacks are viable and discusses techniques for preventing these attacks. The two conditions that enable these attacks are:

1. The cryptographic services of the user's machine must enable the use of a public key in more than one protocol. Although it is not uncommon to assume that a user may have multiple public keys for multiple levels of security, and possibly for multiple job functions;

it has previously not been seen necessary for a user to have a separate public key for every authentication protocol that they use.

2. The second protocol which enables these attacks needs to be installed on the masqueraded user's machine and must have access to cryptographic services on that machine that utilize the user's public-key algorithms. As pointed out in the third example, this protocol need not be a specific tailored protocol, but can be an authenticated secure protocol that happens to share a message format with the attacked protocol.

Given that these conditions may exist in a wide range of systems and implementations of the PKI, we need to explore methods that prevent this type of attack. If we make either condition unattainable, we will succeed in blocking these attacks. The remainder of this section discusses how we can thwart these attacks.

4.1 Kelsey, et. al. Design Principles

Kelsey, Schneier, and Wagner [KSW97] present the following five design principles for protocols that "appear to render the chosen protocol attack impossible."

Principle 1. Limit the scope of the key. This addresses the first condition above. However, we need some mechanism to implement this restriction. The current cryptographic API's on the market do not restrict the use of certified keys, or for that matter any keys. As long as the application has a handle for the key, it can use it. The API's have to be designed to securely manage key use.

Principle 2. Uniquely identify each application, protocol, version, and protocol step. This directly maps to condition two above. However, this inclusion of the identifier is not sufficient if the tailored protocol does not follow this restriction, but rather matches the attacked protocol.

Principle 3. Include a fixed unique identifier in a fixed place in the authenticated protocol. As pointed out by Kelsey, Schneier and Wagner, this restriction prevents the multi-protocol attack using protocols designed with this principle. However, it does not prevent any such attack using *tailored* protocols that are not restricted to this design limitation.

Principle 4. Tie the unique identifier to encryption in a way that forces the identifier to be used for successful decryption. This technique prevents a blind signing protocol (one that will automatically run the crypto-algorithm using their private key and then responding with the result) from being used to decrypt secret messages. A more generic approach is to force all encryption algorithms to add a checksum to the original message prior to encryption, and to use it as a verification of decryption. This forces encryption and decryption to be non-symmetric and will prevent this class of attack. Note that we don't specify forcing the implementations to do this, but rather insist that the algorithm specify adding the checksum, ensuring that all implementations will comply.

Principle 5. Include support for these mechanisms in smartcards. This principle is not actually a design principle, but rather a statement that infers that *any* device that implements cryptographic protocols must be able to enforce restrictions that render multi-protocol attacks impossible (or at least restrict their scope).

4.2 Addressing the Design Principles.

The design principles just discussed are not sufficient for stopping multi-protocol attacks unless there is strong underlying support for the mechanisms. To address condition 1, and principle 1, we need to limit use of a public key to a specific set of protocols. This limitation requires one of the following:

- Inclusion of protocol identifiers in the public-key certificate, along with universal protocol numbering and a secure mechanism for a protocol implementation to prove to the system that it indeed implements the specified protocol.
- Cryptographic subsystem support for key limitations, where the user specifies which protocols can use the specified keys. This places a large portion of the burden on the user, who needs to understand the full ramifications of key sharing.
- Use the certified public key to certify new public keys that are generated for a single use (as in [SH97]). The certified public key is then used in only the key distribution protocol, and each new public key is used only once. A multi-protocol attack may still be possible depending on the content and format of the newly distributed public-key certificate. Examination of such attacks is left for future publication.

To address condition 2, and principles 2 and 3, we need to limit installation of all cryptographic protocols. This limitation requires the following:

- Cryptographic subsystem support for protocol validation. Current cryptographic subsystems require some form of authorization for the cryptographic modules (e.g., DES, RSA, etc) and for public-key certificates (e.g., PKS#6), but then allow user applications full access to the cryptographic subroutines. A malicious user application can then use these routines to implement any tailored protocol. Unfortunately, this forces all protocols to be installed within the cryptographic subsystem.
- A default identifier numbering mechanism (as in principles 2 and 3) that will uniquely identify each run of the protocol. Note that this has to be used in conjunction with the cryptographic subsystem support, otherwise a protocol can be tailored to use the same identifiers. As such, these identifiers must be defined by the cryptographic subsystem and not provided by the protocol user.
- Validation that there is no inadvertent flaw between two registered protocols that would lead to a multi-protocol attack. This can be supported by a universal numbering system as discussed above.

Without full consideration of these restrictions and conditions, multi-protocol attacks will be possible. The public-key infrastructure needs to be designed with an end use in mind that includes consideration of multi-protocol attacks and other possible public-key misuses.

5 Conclusions

The public-key infrastructure will be utilized to store and disseminate certified copies of user's public keys for use in secure transmission and digital signature verification. This paper presented a class of attacks, *multi-protocol attacks*, which can be used to break otherwise secure

public-key based authentication algorithms. These attacks are possible when the public-key infrastructure permits the use of a users public key in multiple protocols and when these protocols can be installed on the user's machine.

Some readers may feel that these attacks may not occur since they require the installation of the attacking protocol onto the masqueraded user's machine. The problem is that for everyday user's this is not an unlikely occurrence. A user who wants access to an interesting new network service may be willing to download and install a new authentication protocol. If this protocol is digitally signed and verified to be secure (as with the STS protocol) even cautious users may be willing to install it. For network service providers there is interest in supporting a wide range of protocols to reach the largest number of clients possible.

What has not been addressed in this paper is the security and reliability of the cryptographic service modules on the user's machines. There are several of these being developed a distributed on the network. Not all of them are being developed with the same goals or levels of software engineering quality. In addition, some of them may rely on the underlying operating system security features, an assumption that may not be advisable.

References

[Alv97] J. Alves-Foss. The Failure of Formal Methods in the Analysis of Cryptographic Protocols. *University of Idaho Department of Computer Science*, Technical Report. LAL-97-06, March 1997.

[AN94] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. In *Proc. IEEE Symposium on Research in Security and Privacy*, 122-136, 1994.

[CCITT89] CCITT, Draft Recommendation X.509. *The Directory--Authentication Framework*. Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(26):644-654, November 1996.

[DvOW92] W. Diffie, P. van Oorschot and M. Wiener. Authentication and Authenticated Key Exchanges. *Deisgns, Codes, and Cryptography*, 2:107-125, 1992.

[KSW97] J. Kelsey, B. Schneier and D. Wagner. Protocol Interactions and the Chosen Protocol Attack. In *Proc. Security Protocols - 5th International Workshop,* 91-104, 1997 (LNCS 1361).

[Low96] G. Lowe. Some New Attacks upon Security Protocols. In *Proc. 9th Computer Security Foundations Workshop*, 162-169, 1996.

[NIST93] National Institute of Standards and Technology, NIST FIPS PUB 188. *Secure Hash Standard*. U.S. Department of Commerce, May 1993.

[NS78] R. Needham and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993-999, December 1978.

[RSA93a] RSA Laboratories. *PKCS #6: Extended-Certificate Syntax Standard*. Version 1.5, November 1993.

[RSA93b] RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard*. Version 1.5, November 1993.

[Riv92] R. Rivest. *The MD5 Message Digest Algorithm*. RFC 1321, April 1992.

[SH97] B. Schneier and C. Hall. An Improved E-mail Security Protocol. In *Proc. 13th Annual Computer Security Applications Conference.* ACM Press, 232-238, December 1997.

[SvO96] P. Syverson and P. van Oorschot. A Unified Cryptographic Protocol Logic. *Draft*, December 1996.