4 Informal Analysis Schemes of Cryptographic Protocols

Abstract Four security definitions about unilateral authentication secure, mutual authentication secure, unilateral session key secure, or mutual session key secure are given respectively under the computational model of matching conversation and indistinguishability. An informal analysis approach based on trusted freshness is presented, and the analysis results suggest the correctness of a protocol or the way to construct attacks intuitively from the absence of security properties. Then, the reasons why typical attacks on authentication protocols exist are discussed based on trusted freshness, and corresponding examples are illustrated to corroborate the discussion.

Due to the asynchronous nature of contemporary networks, establishing whether the security of an authentication protocol is adequate or not is far more difficult than one may have initially imagined[1–15]. A variety of useful rigorous ways have been developed for analyzing and reasoning about cryptographic protocols and they have been proved much useful while designing and analyzing a cryptographic protocol[16–21].

However, there are some important issues that still lack satisfactory treatment. (1) What's the efficient way to distinguish whether a message is fresh or not, to prevent replay attacks, parallel attacks and interleaving attacks. For example, Burrows, Abadi and Needham presented the famous BAN logic[7], which states: the formula X has not been sent in a message at any time before the current run of the protocol, then X is fresh. This is subtle, hence there exists an interleaving attack which is also a replay attack on Needham-Schroeder public key protocol (Needham-Schroeder protocol for short) even if the Needham-Schroeder protocol could be proved secure by BAN logic[1–3, 7, 8, 22]. (2) How to avoid the dependency of analysis on the idealization of a protocol, the concrete formalization of attackers' possible behaviors and the formalization of concurrent runs of protocols. (3) What are the precise specifications of the guarantee for the security of an authentication protocol, which prove the correctness of the protocol sufficiently and necessarily.

In this chapter, 4 security definitions about unilateral authentication se-

L. Dong et al., *Cryptographic Protocol*

cure, mutual authentication secure, unilateral session key secure, or mutual session key secure are given respectively under the computational model of matching conversation and indistinguishability. The presented conditions to guarantee the security adequacy of unilateral entity authentication protocols, mutual entity authentication protocols, unilateral key establishment protocols and mutual key establishment protocols are proved. A novel idea of protocol security analysis is presented based on trusted freshness, which is called the freshness principle. Security analysis based on trusted freshness can efficiently distinguish whether a message is fresh or not, and the analysis results suggest the correctness of a protocol or the way to construct attacks intuitively from the absence of security properties. The reasons why typical attacks on authentication protocols exist are discussed based on trusted freshness, and corresponding examples are illustrated to corroborate the discussion.

4.1 The security of cryptographic protocols

Recall the security assumptions of the cryptographic protocols. Suppose there exists a probabilistic polynomial time (PPT) attacker I that has full control of the communication links as described in Dolev-Yao threat model[23]. Besides this, suppose that the Dolev-Yao attacker I can also launch the Adaptive Chosen Ciphertext Attacks (CCA2) without limitations. Suppose the cryptographic primitives are secure against Indistinguishable Adaptive Chosen Ciphertext Attack (IND-CCA2). That is, in IND-CCA2 security strength, the failures in cryptographic protocols are not in any way related to the strength or weakness of a particular cryptographic primitive used (that is, the cryptographic primitives are perfect in this attack model), but related to the protocol logic flaws, which permit the attacker to break the security goals of cryptographic protocols without necessarily breaking the particular cryptographic primitives used. Suppose that a legitimate party is either totally corrupted or totally secure. Suppose that each participant has his own private key and all other parties' public keys (respectively, the shared long-term key between co-operative principals or trusted third parties) in public-key case (respectively, in symmetric-key case), which are deployed safely before the cryptographic protocol run via non-cryptographic, and out-of-band techniques. Furthermore, private keys and shared keys are commonly assumed to be too long to guess in a computationally feasible way. In general, an authentication protocol is considered flawed if a principal concludes a normal run of the protocol with its intended communication partners while the intended partner would have a different conclusion. This book mainly discusses Challenge-Response authentication protocols.

4.1.1 Authenticity and confidentiality under computational model

The security definition under computational model provides a high confidence of the security of a cryptosystem.

Definition 4.1 A conversation is a sequence of timely ordered messages that a participant sent out (respectively, received), and as consequent responses, received (respectively, sent). Let $\tau_1 < \tau_2 < \cdots \tau_n$ be a time sequence recorded by the participant when it converses. The conversation can be denoted by the following sequence: $conv = (\tau_1, m_1, m_1'), (\tau_2, m_2, m_2'), \cdots, (\tau_n, m_n, m_n')$[17, 24].

This sequence encodes that at time τ_1, the participant was asked m_1 and responded with m_1', and then, at some later time $\tau_2 > \tau_1$, the participant was asked m_2, and responded with m_2', and so on, until, finally, at time τ_n it was asked m_n, and responded with m_n'. If $m_1 =$"", that is no message input, then the participant is the initiator, otherwise, it is called the responder. If $m_n =$"", there is no message output, then the participant ends the conversation. At the end of a protocol run, each participant makes a decision about the authentication of the intended partner: accept, reject, or is undetermined.

Definition 4.2 Suppose there exists a cryptographic protocol run between principals A and B. Let $conv = (\tau_0, "", m_1), (\tau_2, m_1', m_2), (\tau_4, m_2', m_3), \cdots, (\tau_{2(t-1)}, m_{t-1}', m_t)$ be a conversation of A. It is said that B has a conversation $conv'$ which matches $conv$ if there exists time sequence $\tau_1 < \tau_2 < \cdots \tau_n$ and $conv' = (\tau_1, m_1, m_1'), (\tau_3, m_2, m_2'), (\tau_5, m_3, m_3'), \cdots, (\tau_{2t-1}, m_t, m_t')$ where $m_t =$"no message output". These two conversations are called matching conversations[17, 22, 24, 25].

Given a protocol Π between principals A and B, if a principal like A (or B) with a conversation $conv$ believes that B (or A) always has a conversation $conv'$ which matches $conv$ whenever it is allowed to complete a protocol run, then this authentication protocol is secure from the point of view of A (or B). Here the attacker wins if the principal A or B has reached "accept" decision while A or B does not have a matching conversation in B or A.

We follow the probabilistic indistinguishability definitional approach[25] presented by Goldwasser and Micali to define confidentiality security. Here, that the attacker has broken the scheme means that: without breaking any cryptographic primitive and knowing the corresponding key, the attacker can still learn something about the established new session key under the run of a cryptographic protocol. Here "learn" is defined as distinguishing the value of a key generated by the cryptographic protocol from an independent randomly chosen key.

4.1.2 Security definitions

Based on the security definition of authenticity, the Unilateral entity Authentication Secure definition (UA-Secure) and Mutual entity Authentication Secure definition (MA-Secure) are presented; based on the security definition of authenticity and confidentiality, the Unilateral authenticated Key Secure (UK-Secure) and Mutual authenticated Key Secure (MK-Secure) are presented[19].

Definition 4.3 (UA-Secure) Unilateral entity Authentication Secure: an authentication protocol Π is called UA-Secure from the point of view of A if the attacker cannot win with a non-negligible probability for any attacker I in Dolev-Yao threat model with the assistance of cryptanalysis training course. Here the attacker wins if the principal A has reached "accept" decision while A does not have a matching conversation in B.

Actually, according to the notion matching conversation, we can only prove that the intended principal has correctly responded to the challenge of the current protocol run, that is, the intended principal is present, but we can not prove that the intended principal has participated in this protocol run where the challenge is generated. Hence, there exists Man-in-the-Middle attack even the presence of the intended principal has been proved. To correct this problem, the challenge should be associated with all the participants in the protocol, then the identity of the intended principal could not be authenticated.

Definition 4.4 (MA-Secure) Mutual entity Authentication Secure: an authentication protocol Π is called MA-Secure if the attacker cannot win with a non-negligible probability for any attacker I in Dolev-Yao threat model with the assistance of cryptanalysis training course. Here the attacker wins if any principal A or B has reached "accept" decision while A or B does not have a matching conversation in B or A.

Definition 4.5 (UK-Secure) Unilateral authenticated Key Secure: let k be the value of the corresponding new session key. We toss a coin b, $b \xleftarrow{R} \{0,1\}$. If $b = 0$, we provide the attacker I with the value k. Otherwise we provide the attacker I with a value r randomly chosen from the probability distribution of keys generated by protocol Π. At the end of its run, the attacker I outputs a bit b' (as its guess for b). An authentication protocol Π is called UK-Secure from the point of view of A if the following properties hold for any attacker I in Dolev-Yao threat model with the assistance of cryptanalysis training course:

1) If uncorrupted party A believes that A has completed a session with the intended opposite party B, then A trusts that the uncorrupted party B must have responded to the same session, and they both output the same key k.

2) The probability that the attacker I guesses correctly the bit b (i.e., outputs $b' = b$) is no more than $1/2$ plus a negligible fraction in the security parameter.

Definition 4.6 (MK-Secure) Mutual authenticated Key Secure (SK-Secure[19]): let k be the value of the corresponding session key. We toss a coin b, $b \xleftarrow{R} \{0,1\}$. If $b = 0$, we provide the attacker I with the value k. Otherwise we provide the attacker I with a value r randomly chosen from the probability distribution of keys generated by protocol Π. At the end of its run, the attacker I outputs a bit b' (as its guess for b). An authentication protocol Π is called MK-Secure if the following properties hold for any attacker I in Dolev-Yao threat model with the assistance of cryptanalysis training course:

1) Protocol Π satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key.

2) The probability that the attacker I guesses correctly the bit b (i.e., outputs $b' = b$) is no more than $1/2$ plus a negligible fraction in the security parameter.

Recall that each party creates and maintains a local state for a particular protocol run until a session ends its run. To corrupt a party means that the attacker learns all the internal memory of that party including long-term secrets (such as private keys or master shared keys used across different sessions) and session-specific information contained in the party's memory (such as internal state of incomplete sessions and session-keys corresponding to completed sessions).

Mutual authenticated key secure includes mutual authenticated key transport secure (for mutual authenticated key transport) and mutual authenticated key agreement secure (for mutual authenticated key agreement).

4.2 Security mechanism based on trusted freshenss

A novel idea of protocol security analysis based on trusted freshness will be presented in this section[26]. The presentations include the freshness principle, which is the key of the security analysis based on trusted freshness; the substantial and necessary requirements to meet four computational security definitions; a manual analysis method based on the freshness principle, and an illustration of the analysis method based on trusted freshness.

4.2.1 Notions

Some notions used in the security analysis method based on trusted freshness will be given in this subsection.

Definition 4.7 (Freshness) From the point of view of a participant in a protocol run, freshness means that a freshness identifier or a message is confirmed to be new for a particular run of the protocol. If a freshness identifier is new generated by the principal itself for this run or a message is conveyed with this new generated identifier via a one-way transformation or a trapdoor one-way transformation, then the freshness of the freshness identifier or the message is confirmed.

Definition 4.8 (Trusted freshness) A trusted freshness identifier (also called trusted freshness) is a freshness identifier whose freshness has been confirmed via generation of the principal itself or via a one-way transformation or a trapdoor one-way transformation including a trusted freshness identifier. The trusted freshness identifiers include trusted nonces, trusted timestamps, trusted session keys or trusted shared parts of a session key. Note that the trusted freshness identifiers are different for different protocol runs, are different for each participant in the same protocol run, and are different for the same principal in different protocol runs.

Definition 4.9 (Fresh message) From the point of view of a participant in a protocol run, a fresh message is a sent or received message that includes a trusted freshness.

Definition 4.10 (Term) A term $\hat{m}$ is a fresh message owned by a principal that may be exchanged in a particular protocol run. A term set $\hat{M}$ is the collection of all terms in a protocol run. Terms can be recursively defined as:

1) If $\hat{m}$ is a trusted freshness identifier, then $\hat{m}$ is a term.

2) If $\hat{m}$ is a term, o is a principal identity or a freshness identifier, and $\{\hat{m}, o\}$, or $\{o, \hat{m}\}$ is encrypted or should be decrypted via a one-way transformation or a trapped door one-way transformation, then $\{\hat{m}, o\}$, or $\{o, \hat{m}\}$ is a term. $\{\hat{m}, o\}$ and $\{o, \hat{m}\}$ are regarded as the same term in the security analysis approach based on trusted freshness.

3) If $\hat{m}$ is a term, k is a cryptographic key known by the principal, and $\{\hat{m}\}_k$ is encrypted or should be decrypted via a one-way transformation or a trapped door one-way transformation, then $\{\hat{m}\}_k$ is a term. If $\hat{m}$ is a term, o is a principal identity or a freshness identifier, then $\{o\}_{\hat{m}}$ is a term.

A maximal term is the longest term which is constructed from a single message of a particular protocol run via the applications of Definition 4.12 as many times as possible.

There may be more than one maximal terms in a message, for example, the message 4 in a protocol:

$$\text{Message 4} \quad A \to B : \{A, k_{AB}, T_B\}_{K_{BS}}, \{N_B\}_{k_{AB}}$$

Both $\{A, k_{AB}, T_B\}_{K_{BS}}$ and $\{N_B\}_{k_{AB}}$ are the maximal term of the message.

If the maximal term of a message in a protocol run is the same as that of another message of this protocol run, then we say this protocol has similar term in these two messages.

That is, one single message's maximal term should not be the same as another single message's maximal term in a particular protocol run. Otherwise, one of these two messages may be interleaving replayed in some cases. Recall the similar term in Message 4 and Message 5 in the Clark-Jacob attack on the Woo-Lam protocol of Example 3.28.

A signed term is a binary group $(\delta, \hat{m})$, where δ is a sign, $\hat{m} \in \hat{M}$. A signed term is stated as $+\hat{m}$ or $-\hat{m}$. $+\hat{m}$ and $-\hat{m}$ states a sent out or received fresh message.

Example 4.1 Figure 4.1 illustrates an example of terms. The protocol intends to establish a new session key k_{AB} between A and B, with the help of the trusted server S. N_A, N_B are nonces generated by A and B respectively; K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively.

$$\begin{aligned}
&\text{Message 1} \quad A \to S : A, B, N_A \\
&\text{Message 2} \quad S \to A : \{N_A, k_{AB}, B, \{k_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&\text{Message 3} \quad A \to B : S, \{k_{AB}, A\}_{K_{BS}} \\
&\text{Message 4} \quad B \to A : \{N_B\}_{k_{AB}} \\
&\text{Message 5} \quad A \to B : \{N_B - 1\}_{k_{AB}}
\end{aligned}$$

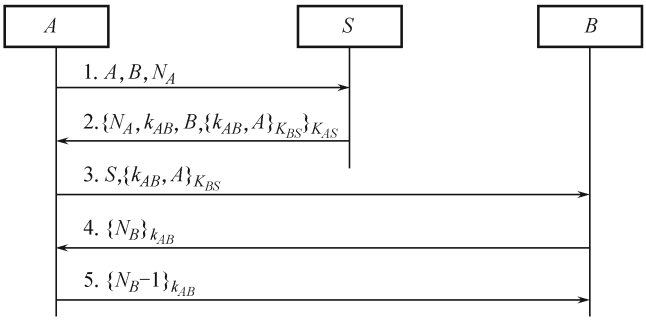

Fig. 4.1 Example of terms.

Suppose the principal B believes that N_B is a trusted freshness identifier (N_B is generated by B in this protocol run). Suppose the principal A believes

that N_A is a trusted freshness identifier (N_A is generated by A in this protocol run).

$$\text{Message 1} \quad A \rightarrow S : A, B, N_A$$

Upon sending Message 1, A has gotten the terms as in Table 4.1, where $\{...N_A...\}$ is the maximal term of Message 1.

Upon receiving Message 1, B has not gotten any terms since there does not exist a trusted freshness identifier for B.

$$\text{Message 2} \quad S \rightarrow A : \{N_A, k_{AB}, B, \{k_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$

Table 4.1 The terms owned by A from Message 1

No.	Term	Signed Term
1	$\{...N_A...\}$	$+\{...N_A...\}$
Maximal	Term	
	$\{...N_A...\}$	

Upon sending Message 2, S has gotten the terms as in Table 4.2. Upon receiving Message 2, A has gotten the terms as in Table 4.3.

Table 4.2 The terms owned by S from Message 2

No.	Term	Signed Term
1	$\{...k_{AB}...\}$	$+\{...k_{AB}...\}$
2	$\{...k_{AB}, A...\}$	$+\{...k_{AB}, A...\}$
3	$\{...k_{AB}...\}_{K_{BS}}$	$+\{...k_{AB}...\}_{K_{BS}}$
4	$\{...k_{AB}, A...\}_{K_{BS}}$	$+\{...k_{AB}, A...\}_{K_{BS}}$
5	$\{...N_A, k_{AB}...\}$	$+\{...N_A, k_{AB}...\}$
6	$\{...k_{AB}, B...\}$	$+\{...k_{AB}, B...\}$
7	$\{...k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$+\{...k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$
8	$\{...N_A, k_{AB}, B...\}$	$+\{...N_A, k_{AB}, B...\}$
9	$\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$+\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$
10	$\{...k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$+\{...k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$
11	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$+\{...N_A, k_{AB}, B, \{...N_A, k_{AB}, A...\}_{K_{BS}}...\}$
12	$\{...N_A, k_{AB}...\}_{K_{AS}}$	$+\{...N_A, k_{AB}...\}_{K_{AS}}$
13	$\{...k_{AB}, B...\}_{K_{AS}}$	$+\{...k_{AB}, B...\}_{K_{AS}}$
14	$\{...k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$+\{...k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
15	$\{...N_A, k_{AB}, B...\}_{K_{AS}}$	$+\{...N_A, k_{AB}, B...\}_{K_{AS}}$
16	$\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$+\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
17	$\{...k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$+\{...k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
18	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$+\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
Maximal	Term	
	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	

Table 4.3 The terms owned by A from Message 2

No.	Term	Signed Term
1	$\{...N_A...\}$	$-\{...N_A...\}$
2	$\{...N_A, k_{AB}...\}$	$-\{...N_A, k_{AB}...\}$
3	$\{...N_A, B...\}$	$-\{...N_A, B...\}$
4	$\{...N_A, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$-\{...N_A, \{...k_{AB}, A...\}_{K_{BS}}...\}$
5	$\{...N_A, k_{AB}, B...\}$	$-\{...N_A, k_{AB}, B...\}$
6	$\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$-\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}$
7	$\{...N_A, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$-\{...N_A, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$
8	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}$	$-\{...N_A, k_{AB}, B, \{...N_A, k_{AB}, A...\}_{K_{BS}}...\}$
9	$\{...N_A, k_{AB}...\}_{K_{AS}}$	$-\{...N_A, k_{AB}...\}_{K_{AS}}$
10	$\{...N_A, B...\}_{K_{AS}}$	$-\{...N_A, B...\}_{K_{AS}}$
11	$\{...N_A, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$-\{...N_A, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
12	$\{...N_A, k_{AB}, B...\}_{K_{AS}}$	$-\{...N_A, k_{AB}, B...\}_{K_{AS}}$
13	$\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$-\{...N_A, k_{AB}, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
14	$\{...N_A, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$-\{...N_A, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
15	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	$-\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$
Maximal	Term	
	$\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$	

Note that from the point of view of A, $\{...N_A, k_{AB}, A...\}_{K_{BS}}$ is the same as a randomly chosen nonce since A does not have possession of the long-term key K_{BS}. Here, $\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$ is the maximal term of Message 2.

As we will show in the following part of this chapter, upon receiving Message 2, A has the belief that k_{AB} is a trusted freshness from a trapdoor one-way transformation $\{...N_A, k_{AB}, B, \{...k_{AB}, A...\}_{K_{BS}}...\}_{K_{AS}}$ including the trusted freshness identifier N_A.

$$\text{Message 3} \quad A \rightarrow B : S, \{k_{AB}, A\}_{K_{BS}}$$

Upon sending Message 3, A has not gotten any terms without the knowledge of the long-term key K_{BS}. Upon receiving Message 3, B has not gotten any terms since there does not exist a trusted freshness identifier for B. So, the maximal term in Message 3 is null.

$$\text{Message 4} \quad B \rightarrow A : \{N_B\}_{k_{AB}}$$

Upon sending Message 4, B has gotten the terms as in Table 4.4.

Table 4.4 The terms owned by B from Message 4

No.	Term	Signed Term
1	$\{...N_B...\}$	$+\{...N_B...\}$
2	$\{...N_B...\}_{k_{AB}}$	$+\{...N_B...\}_{k_{AB}}$
Maximal	Term	
	$\{...N_B...\}_{k_{AB}}$	

Upon receiving Message 4, note that k_{AB} is a trusted freshness, so A has gotten the terms as in Table 4.5.

Table 4.5 The terms owned by A from Message 4

No.	Term	Signed Term
1	$\{...N_B...\}_{k_{AB}}$	$-\{...N_B...\}_{k_{AB}}$
Maximal	Term $\{...N_B...\}_{k_{AB}}$	

So, the maximal term of Message 4 is $\{...N_B...\}_{k_{AB}}$.

$$\text{Message 5} \quad A \rightarrow B : \{N_B - 1\}_{k_{AB}}$$

Upon sending Message 5, note that k_{AB} is a trusted freshness, so A has gotten the terms as in Table 4.6.

Table 4.6 The terms owned by A from Message 5

No.	Term	Signed Term
1	$\{...N_B...\}_{k_{AB}}$	$+\{...N_B...\}_{k_{AB}}$
Maximal	Term $\{...N_B...\}_{k_{AB}}$	

Upon receiving Message 5, B has gotten the terms as in Table 4.7.

Table 4.7 The terms owned by B from Message 5

No.	Term	Signed Term
1	$\{...N_B...\}$	$+\{...N_B...\}$
2	$\{...N_B...\}_{k_{AB}}$	$+\{...N_B...\}_{k_{AB}}$
Maximal	Term $\{...N_B...\}_{k_{AB}}$	

Here, $\{...N_B...\}_{k_{AB}}$ is the maximal term in Message 5.

Example 4.2 Here is an example of similar term. Recall the Example 4.16, the maximal term $\{...N_B...\}_{k_{AB}}$ in Message 4 is the same as that in Message 5, so we say this protocol in Example 4.16 has similar term $\{...N_B...\}_{k_{AB}}$ in Message 4 and Message 5.

Definition 4.11 (Liveness of a principal) From the point of view of a participant in a protocol run, the intended opposite participant is in lively correspondence with him in this session.

Note From the point of view of a participant in a protocol run, the intended opposite participant is specially in lively correspondence with this origin participant in this session. The liveness of a principal with origin is also called origin liveness of a principal.

In lively correspondence with a origin participant means that the origin participant has corroborative evidence that the intended opposite participant is in lively correspondence with this origin participant but not any other participant.

Definition 4.12 (Confidentiality of a freshness identifier From the point of view of a participant in a protocol run, the freshness identifier is transmitted in the form of an encryption that cannot be decrypted by the attacker.

If the freshness identifier is transmitted in the form of a plaintext or an encryption that may be decrypted by the attacker, then the freshness identifier is open. Note that the signature of a freshness identifier is not confidential.

Definition 4.13 (Freshness of a freshness identifier) From the point of view of a participant in a protocol run, the freshness identifier is new generated for this particular protocol run, not an old one or a compromised one.

A principal believes the freshness of the freshness identifier generated by the principal itself.

Definition 4.14 (Association of a freshness identifier) From the point of view of a participant in a protocol run, the freshness identifier is bound to some legitimate participants of this particular protocol run.

In the security analysis of a cryptographic protocol based on trusted freshness, the security properties are described by beliefs, which are the beliefs about the security of a cryptographic protocol owned by each participant in a particular protocol run. The beliefs are about liveness of principal, confidentiality of a freshness identifier, freshness of a freshness identifier and association of a freshness identifier.

4.2.2 Freshness principle

Definition 4.15 (Freshness Principle) For each participant of a cryptographic protocol, the security of the protocol depends only on the sent or received "loose" one-way transformation of a message which includes a trusted freshness.

The security goals are the security objects at the end of the protocol run. The set of security goals constructs the security properties of a cryptographic protocol to achieve.

In practice, a one-way transformation $[M]_k$ can be realized by a pair $(M,prf_k(M))$ where prf_k denotes a keyed pseudorandom function (e.g., a message authentication code in cipher-block-chaining mode of operation, CBC-MAC, or a keyed cryptographic hash function, HMAC) for the case of symmetric technique realization, or a digital signature algorithm for the case of

asymmetric technique realization. These are practically efficient realization[6]. These practical one-way transformations are indeed trapped door one-way transformations for most cases, hence we use "loose" one-way transformation to refer to them. As we have stated in the last chapter, when we refer to "one-way transformation", we usually mean "loose one-way transformation".

Let A, B be the participants of an authentication protocol, we have the following lemmas:

Lemma 4.1 (Liveness Lemma) The liveness of a principal B can be achieved by a participant A via a sent or received "loose" one-way transformation that includes a trusted freshness identifier owned by A, where the "loose" one-way transformation can only be accomplished by the principal B.

Lemma 4.2 (Confidentiality Lemma) The confidentiality of a freshness identifier can be achieved by a participant A if the identifier is transmitted in the form of an encryption that cannot be decrypted by the attacker; if the freshness identifier is transmitted in the form of a plaintext or an encryption that may be decrypted by the attacker, then the freshness identifier is open.

Lemma 4.3 (Freshness Lemma) The freshness of a freshness identifier can be achieved by a participant A via a sent or received "loose" one-way transformation that includes:

1) a new freshness identifier which is generated by the principal A itself;

2) a new freshness identifier which is bound together with A's another trusted freshness identifier in a "loose" one-way transformation, where the "loose" one-way transformation can only be accomplished by the intended participant B.

Lemma 4.4 (Association Lemma) The association of a freshness identifier can be achieved by a participant A via a sent or received "loose" one-way transformation that includes a trusted freshness identifier owned by A, where the "loose" one-way transformation can only be accomplished by the intended principal B, or the identity of the participant is explicitly stated in the "loose" one-way transformation.

Lemma 4.5 (Origin liveness Lemma) The liveness of a principal B with origin can be achieved by a participant A via a sent or received "loose" one-way transformation that includes a trusted freshness identifier owned by A, where the "loose" one-way transformation can only be accomplished by the principal B specially for the origin participant A.

In security analysis approach based on trusted freshness, only "loose" one-way transformation that includes a trusted freshness identifier is considered as an efficient message of a conversation, so terms could be deduced from the transformation. That is to say, only the fresh messages are concerned in security analysis based on trusted freshness, and the message parts that do not contribute to the protocol security property analysis in the trusted

freshness method are omitted.

4.2.3 Security of authentication protocol

Suppose there exists a protocol Π between A and B, the security goal of Π is to authenticate the liveness of a principal entity or establish a new session key to build a secure channel in an insecure network. The new session key k_{ab} can either be generated by any of the authenticated participants or a trusted third party S, or be the output of a function of all protocol participants' random input like N_A and N_B.

Table 4.8 lists the security property requirements to guarantee the security goals of a cryptographic protocol, and we will show that the listed security

Table 4.8 The guarantee of the security adequacy of a cryptographic protocol

Security Goals	Security Properties Achieved by A					Security Properties Achieved by B				
	B	S	N_A	N_B	k_{AB}	A	S	N_A	N_B	k_{AB}
UA-secure (Authenticate B)	1[a]									
UA-secure (Authenticate A)						1				
MA-secure (Authenticate both)	1					1				
Origin UA-secure (Origin Authenticate B)	A1[b]									
Origin UA-secure (Origin Authenticate A)						B1				
Origin MA-secure (Origin Authenticate both)	A1					B1				
UK-secure (Authenticate B)	1				1[c]1[d]AB[e]					
UK-secure (Authenticate A)						1				11AB
MK-secure (Key Transport)	1				11AB	1				11AB
MK-secure (Key Transport)	1		11AB	11AB		1		11AB	11AB	

[a] "1" or "" means that the liveness of the principal is authenticated or unknown respectively.

[b] "A1" or "" means that the liveness of the principal is authenticated with origin or unknown respectively.

[c] "?" or "" means that the confidentiality of the freshness identifier is unknown; "1" means that the confidentiality of the freshness identifier is confidential; "0" means that the freshness identifier is open, which is a plaintext, or a compromised one or an old one.

[d] "?" or "" means that the freshness of the freshness identifier is unknown; "1" means that the freshness identifier is fresh.

[e] "#" or "" means that the freshness identifier is not associated with any principals; "11A" (or "11B") means that the freshness identifier is associated with the principal A (or B); "11AB" or "11BA" means that the freshness identifier is associated with both A and B.

goals are not only necessary but also substantial. Here, when an authentication protocol is MK-secure is proved, and other similar proofs of UA-secure, UK-secure, MK-secure are omitted for interest of concision.

The security property requirements of some cryptographic protocols are given in [26]. And they are:

Theorem 4.1 An authentication-only protocol Π is called UA-Secure if and only if a participant like A believes the liveness of the intended opposite principal B.

Theorem 4.2 An authentication-only protocol Π is called MA-Secure if and only if each participant believes the liveness of both communication principals.

Theorem 4.3 A key establishment authentication protocol Π is called UK-Secure if and only if a participant like A believes the liveness of the intended opposite principal B, and believes the confidentiality, the freshness and also the association of the new session key k with the principal A and the principal B.

Theorem 4.4 A key establishment authentication protocol Π is called MK-Secure if and only if each participant like A (or B) believes the liveness of the intended opposite principal B (or A), and believes the confidentiality, the freshness and also the association of the new session key k with the principal A and the principal B.

Theorem 4.5 A authentication-only protocol Π is called Origin UA-Secure if and only if a participant like A believes the liveness of the intended opposite principal B with origin.

Theorem 4.6 A authentication-only protocol Π is called Origin MA-Secure if and only if each participant believes the liveness of both communication principals with origin.

Here, the liveness of the principal is able to determine the true identity of the other(s) which could possibly gain access to the resulting key; the confidentiality of the new session key k implies secrecy of the key; the freshness of the key k requires that the key is new for this protocol run and it could not be a replay of a compromised one; the association of the k implies preclusion of any unauthorized additional parties from deducing the same key.

Note that the liveness of the intended opposite principal differs subtly, but in a very important manner, from the association of the new session key k. The liveness of a principal implies entity authentication, and an actual communication has been established with such party (or parties); the association of the new session key k is knowledge of the identity of parties which may gain access to the key, rather than corroboration of the entity authentication.

4.2.4 Manual analysis based on trusted freshness

Based on the freshness principle and the accurately presented security goals, an analysis method as shown in Fig. 4.2 based on trusted freshness will be presented, which can be accomplished easily even by hand.

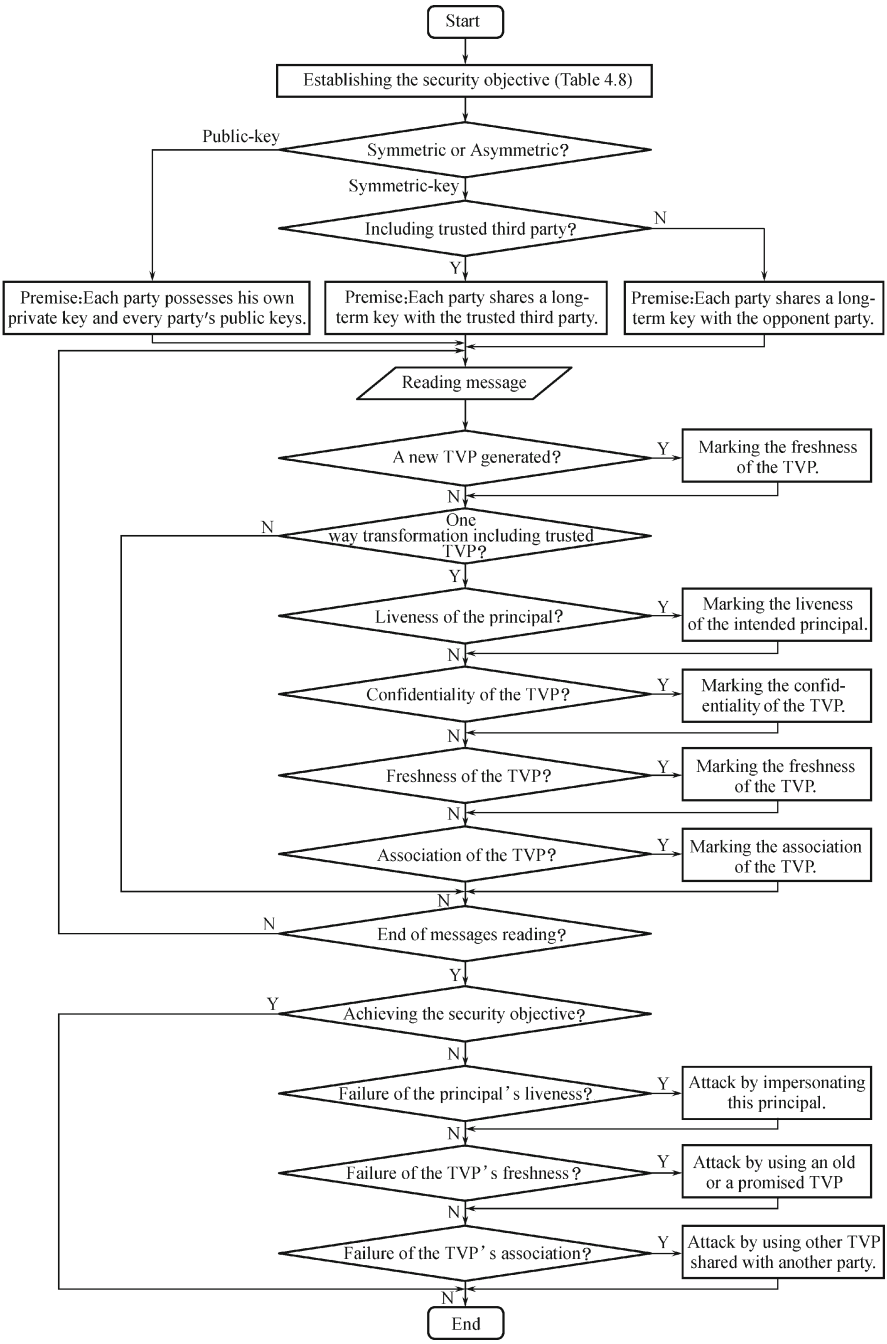

Fig. 4.2 Security analysis of cryptographic protocol based on trusted freshness.

The manual freshness analysis method is refined as follows for the same authentication protocol Π in Section 4.2.3.

1) The security goals to be reached is specified based on Table 4.8.

2) Table 4.9 specifies the premise before the start of the protocol.

Recall that each participant has his own private key and all other parties' public-keys (respectively, the shared long-term key between co-operative principals or trusted third parties) in public-key case (respectively, in symmetric-key case).

Table 4.9 The premise of a cryptographic protocol

Cryptographic schemes	Premise knowledge of A	Premise knowledge of B	Premise knowledge of I
public-key	K_A^{-1}, K_A, K_B, K_I	K_B^{-1}, K_B, K_A, K_I	K_I^{-1}, K_A, K_B, K_I
symmetric-key without TTP	K_{AB}	K_{AB}	
symmetric-key with TTP	K_{AS}	K_{BS}	

3) From the point of view of each legitimate participant, the security properties of a cryptographic protocol are established based on the freshness principle and Lemmas 4.1, 4.2, 4.3, and 4.4 while sending or receiving a "loose" one-way transformation that includes a trusted freshness.

4) Comparing with the security goals established in step 1, the analysis results can either establish the correctness of the protocol when it is in fact correct, or identify the absence of the security properties and the structure to construct attacks based on the absence. From the absence of the security properties of an authentication protocol, various attacks could be directly constructed:

(1) Absence of the liveness of a principal like A: impersonate A to launch an attack, e.g., Otway-Rees protocol[27], Woo-Lam protocol[5, 10].

Absence of the origin liveness of a principal like A: impersonate A by replaying the corroborative evidence from A which may be generated for any other participants but the original participant B.

(2) Absence of the freshness of a freshness identifier: launch an attack by replaying the recorded one-way transformation with a compromised session key, e.g., Needham-Schroeder shared key protocol[9].

(3) Absence of the association of a freshness identifier: launch an attack by confusing a legitimate principal like B to believe a session key k' between I and A (or B) to be the key between A and B, e.g., the Needham-Schroeder public-key protocol[9].

4.2.5 Application of security analysis based on trusted freshness

Example 4.3 Recall the analysis of Needham-Schroeder public-key protocol in Example 3.16 and Example 3.20, Fig. 4.3 illustrates the analysis pro-

cedure of the Needham-Schroeder public-key protocol based on the trusted freshness analysis method.

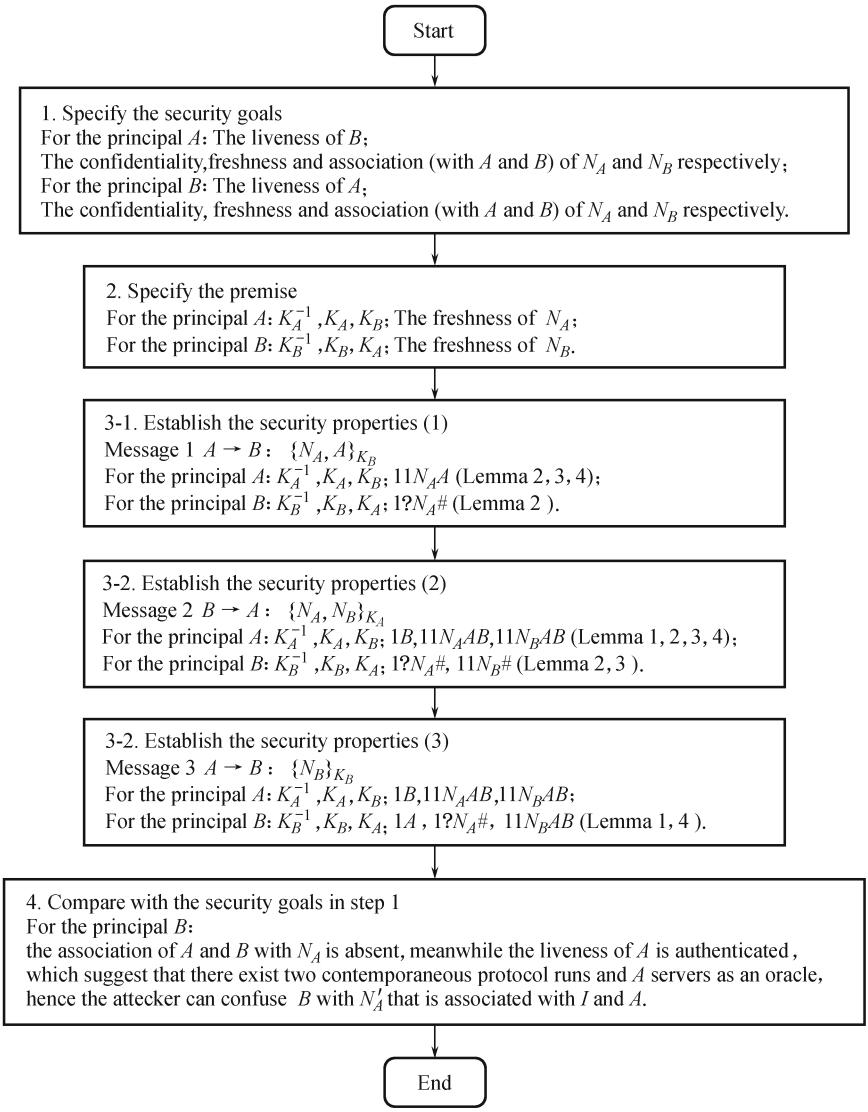

Fig. 4.3 Security analysis of Needham-Schroeder public-key protocol based on trusted freshness.

By analyzing, we get result on security properties, as shown in Tables 4.10, 4.11, and 4.12.

Table 4.10 Security properties achieved by A in the Needham-Schroeder public-key protocol

	Presence of B	N_A			N_B		
		Confidentiality	*Freshness*	*Association*	*Conf.*	*Fresh.*	*Asso.*
Message 1		1	1	A			
Message 2	1			AB	1	1	AB
Message 3							
End of run	1	11AB			11AB		

Table 4.11 Security properties achieved by B in the Needham-Schroeder public-key protocol

	Presence of A	N_A			N_B		
		Confidentiality	*Freshness*	*Association*	*Conf.*	*Fresh.*	*Asso.*
Message 1		1	?	#			
Message 2					1	1	#
Message 3	1						AB
End of run	1	1?#			11AB		

For the sake of ease, the security properties achieved by A and B are simplified as in Table 4.12.

Table 4.12 Security analysis of the Needham-Schroeder public key protocol

	A			B		
	B	N_A	N_B	A	N_A	N_B
Message 1		11A			1?#	
Message 2	1	11AB	11AB			11#
Message 3				1		11AB
End of run	1	11AB	11AB	1	1?#	11AB

4.3 Analysis of classic attacks

A successful attack on an authentication or key establishment protocol usually does not refer to breaking a cryptographic algorithm, e.g., via a complexity theory-based cryptanalysis technique. Instead, it usually refers to Malice's unauthorized and undetected acquisition of a cryptographic credential or nullification of a cryptographic service without breaking a cryptographic algorithm. It actually does not require very sophisticated techniques for an adversary to mount these attacks on a lower-layer communication protocol, as we have seen in Example 1.3.

Over several years, many different types of attacks on cryptographic primitives and protocols have been identified, and it is impossible for us to know all the protocol attacking techniques an adversary may use since the adversary will constantly devise new techniques. In this section, several typical attacks on cryptographic protocols will be analyzed, and the reasons why

these protocols are flawed will be discussed based on trusted freshness to provide us with insight into how to develop stronger protocols. Notice that an adversary may actually launch attacks via a combined way of the listed well-known protocol attacking techniques[22].

In the specific examples below, A and B are the legitimate parties Alice and Bob, and I is the adversary Malice who could also be a legitimate participant in some cases.

4.3.1 Man in the middle attack

In man-in-the-middle attack (often abbreviated to MITM), the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection while in fact the entire conversation is controlled by the attacker. A man-in-the-middle attack can only be successful when the attacker can impersonate each entity to the satisfaction of the other. Most cryptographic protocols include some form of entity authentication specifically to prevent MITM attacks. In essence, man-in-the-middle attack is generally applicable to a communication protocol where mutual entity authentication is absent.

Example 4.4 Diffie-Hellman key agreement[28] provides the first practical solution to the key distribution problem, allowing two parties, never having met in advance or having shared keying material, to establish a shared secret by exchanging messages over an insecure network, as shown in Fig. 4.4. The security rests on the intractability of the Diffie-Hellman problem and the related problem of computing discrete logarithms.

$$\begin{aligned} &\text{Message 1} \quad A \rightarrow B : \alpha^x \bmod p \\ &\text{Message 2} \quad B \rightarrow A : \alpha^y \bmod p \end{aligned}$$

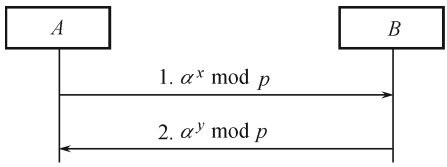

Fig. 4.4 The basic version of Diffie-Hellman key agreement protocol.

Notation

A and B are two protocol principals, x and y are randomly chosen as their private keys by A and B, respectively.

Premise

An appropriate large prime p and a generator element α of $\mathbb{Z}_p^*$ ($2 \leqslant \alpha \leqslant p-2$) are selected and published. x and y are randomly chosen hence could not be found out.

Protocol actions

1) In Message 1, A randomly chooses a secret "x, $1 \leqslant x \leqslant p-2$", and sends B message "$\alpha^x \bmod p$".

2) Upon receiving Message 1, B gets "α^x" and computes the shared key as "$k = (\alpha^x)^y \bmod p = \alpha^{xy} \bmod p$".

3) In Message 2, B randomly chooses a secret "y, $1 \leqslant y \leqslant p-2$", and sends A message "$\alpha^y \bmod p$".

4) Upon receiving Message 2, A gets "α^y" and computes the shared key as "$k = (\alpha^y)^x \bmod p = \alpha^{yx} \bmod p$".

Note that "$\alpha^{xy} \bmod p = \alpha^{yx} \bmod p$", hence A and B have computed the same key k. This is how the Diffie-Hellman key exchange protocol achieves a shared key between two communication parties.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the TVP x; from Lemma 4.2, B has the confidentiality assurance of the TVP x, but B could not deduce the freshness of the TVP x.

2) In Message 2, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and freshness assurances of the TVP y; from Lemma 4.2, A has the confidentiality assurance of the TVP y, but A could not deduce the freshness of the TVP y.

3) At the end of the protocol run, A and B have computed the same key "$k = \alpha^{xy} \bmod p$", and have gotten the confidentiality and freshness assurances of k from the confidentiality and freshness of the TVP x and the TVP y respectively. However, the whole transmitted messages in this protocol could not provide the assurance of the association of A with k for B, and B with k for A.

The analyzing result is indicated in Table 4.13.

Table 4.13 Security analysis of the Diffie-Hellman key agreement protocol

	A				B			
	B	x	y	k	A	x	y	k
Message 1		11#				1?#		
Message 2			1?#				11#	
End of run		11#	1?#	11A		1?#	11#	11B

This basic Diffie-Hellman protocol version provides none authentication, entity authentication and key confirmation, hence it can only provide secrecy protection of the resulting key from eavesdroppers, but not from active

adversaries. Fig. 4.5 illustrates the "man-in-the-middle" attack on the basic unauthenticated Diffie-Hellman key establishment protocol.

$$
\begin{array}{ll}
\text{Message 1} & A \rightarrow I(B) : \alpha^x \bmod p \\
\text{Message 1}' & I(A) \rightarrow B : \alpha^{x'} \bmod p \\
\text{Message 2}' & B \rightarrow I(A) : \alpha^y \bmod p \\
\text{Message 2} & I(B) \rightarrow A : \alpha^{y'} \bmod p
\end{array}
$$

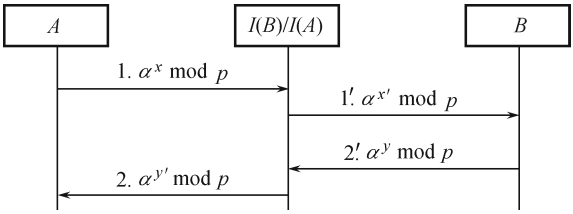

Fig. 4.5 The attack on the basic Diffie-Hellman key agreement.

Protocol actions

1) In Message 1, A randomly chooses a secret "x, $1 \leqslant x \leqslant p-2$", and sends B message "$\alpha^x \bmod p$".

2) The adversary I intercepts A's exponential α^x and replaces it with $\alpha^{x'}$ where x' is a secret chosen by I; meanwhile I intercepts B's exponential α^y and replaces it with $\alpha^{y'}$ where y' is a secret chosen by I.

3) At the end of protocol run, A forms session key "$k_1 = (\alpha^{y'})^x \bmod p = \alpha^{y'x} \bmod p = \alpha^{xy'} \bmod p$", and B forms session key "$k_2 = (\alpha^{x'})^y \bmod p = \alpha^{x'y} \bmod p$", while I can compute both keys k_1 and k_2.

4) When A subsequently sends a message to B encrypted under k_1, I decrypts it, re-encrypts the plaintext under k_2, and forwards it to B. Similarly, I decrypts the message encrypted by B (for A) under k_2, and re-encrypts it under k_1. Both A and B believe that they communicate securely, while I can read all traffic.

4.3.2 Source-substitution attack

In source-substitution attack, the attacker makes a substitution of a source entity identity with the identity of the adversary, making the victim believe that it is talking directly to the intended entity while in fact the entire conversation is controlled by the adversary. e.g., in Example 4.5, the adversary registers source entity's public-key as its own. A source-substitution attack can only be successful when the adversary can impersonate an entity to the satisfaction of the other. In essence, source-substitution attack is generally applicable to a communication protocol where entity authentication is absent.

Example 4.5 The ElGamal key agreement is a Diffie-Hellman variant providing a one-pass protocol with unilateral key authentication[29]. This protocol is more simply Diffie-Hellman key agreement wherein the public exponential of the recipient is fixed and has verifiable authenticity, as shown in Fig. 4.6.

Message 1 $A \rightarrow B : \alpha^x \bmod p$

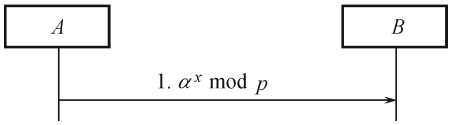

Fig. 4.6 ElGamal key agreement in one-pass.

Notation

A and B are two protocol principals. x is randomly chosen as its private key by A. b is a preselected secret random integer.

Premise

An appropriate large prime p and a generator element α of $\mathbb{Z}_p^*$ ($2 \leqslant \alpha \leqslant p-2$) are selected and published. The public-key of the recipient is known to the originator.

Protocol actions

1) One-time setup (public-key generation and publication). B picks an appropriate large prime p and a generator element "α of $\mathbb{Z}_p^*$", selects a random integer "b, $2 \leqslant b \leqslant p-2$", and computes "$\alpha^b \bmod p$", then B publishes p, α and α^b, keeping private key b secret.

2) Each time a shared key is required.

(1) A obtains an authentic copy of B's public-key α^b. A randomly chooses a secret "$x, 1 \leqslant x \leqslant p-2$", and sends B message "$\alpha^x \bmod p$".

(2) A computes the shared key as "$k = (\alpha^b)^x \bmod p = \alpha^{bx} \bmod p = \alpha^{xb} \bmod p$"; B receives α^x and computes the same shared key as "$k = (\alpha^x)^b \bmod p = \alpha^{xb} \bmod p$".

Protocol security analysis

1) Before the protocol run, suppose A and B have the assurance that b is a confidential and long-term key which is known only by B.

2) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the TVP x; from Lemma 4.2, B has the confidentiality assurance of the TVP x, but B could not deduce the freshness of the TVP x.

3) At the end of the protocol run, A has computed the key $k = \alpha^{xb} \bmod p$. From Lemma 4.2, Lemma 4.3 and Lemma 4.4, A has gotten the confidentiality and freshness assurances of k from the confidentiality and freshness of the

TVP x, and A has gotten the association of A and B with k from the TVP x and B's long-term key b respectively. From Lemma 4.2, B has the confidentiality and the association (B with k) assurances, from B's long-term private key b, but B could not deduce the freshness of k and also the association assurance of A with k.

The analyzing result is indicated in Table 4.14. The recipient B in this protocol has neither corroboration that it shares the secret key k, nor any key freshness assurance. Neither party obtains entity authentication or key confirmation. The adversary can launch an attack by impersonating A directly.

Table 4.14 Security analysis of the ElGamal key agreement in one-pass

	A				B			
	B	x	b	k	A	x	b	k
Message 1		11#	11B			1?#	11B	
End of run		11#	11B	11AB		1?#	11B	1?B

Example 4.6 The MTI two-pass key agreement protocol as shown in Fig. 4.7 is a variant of Diffie-Hellman key agreement to yield time-variant session key with mutual key authentication against passive attacks[28–30].

$$\text{Message 1} \quad A \to B : \alpha^x \bmod p$$
$$\text{Message 2} \quad B \to A : \alpha^y \bmod p$$

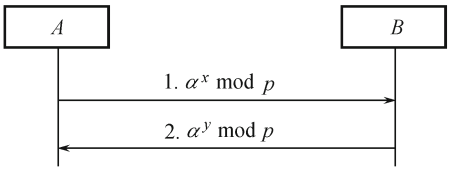

Fig. 4.7 The MTI key agreement protocol in two-pass.

Notation

A and B are two protocol principals, a ($1 \leqslant a \leqslant p-2$) and b ($1 \leqslant b \leqslant p-2$) are randomly chosen integers as its long-term private keys by A and B respectively.

Premise

An appropriate large prime p and a generator element α of "$\mathbb{Z}_p^*$ ($2 \leqslant \alpha \leqslant p-2$)" are selected and published. A has the assurance that only B knows the corresponding long-term private key b of "$\alpha^b \bmod p$", B has the assurance that only A knows the corresponding long-term private key a of "$\alpha^a \bmod p$".

Protocol actions

1) In Message 1, A randomly chooses a secret "x, $1 \leqslant x \leqslant p-2$", and sends B message "$\alpha^x \bmod p$".

2) In Message 2, B randomly chooses a secret "y, $1 \leqslant y \leqslant p-2$", and sends A message "$\alpha^y \bmod p$".

3) B receives "$\alpha^x \bmod p$" and computes the shared key as "$k = (\alpha^x)^b(\alpha^a)^y = \alpha^{bx+ay} \bmod p$".

4) A receives α^y and computes the shared key as "$k = (\alpha^y)^a(\alpha^b)^x = \alpha^{ay+bx} = \alpha^{bx+ay} \bmod p$".

Protocol security analysis

1) Before the protocol run, suppose A and B have the assurance that a is a confidential and long-term key which is known only by A, and that b is a confidential and long-term key which is known only by B.

2) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the TVP x; from Lemma 4.2, B has the confidentiality assurance of the TVP x, but B could not deduce the freshness of the TVP x.

3) In Message 2, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and freshness assurances of the TVP y; from Lemma 4.2, A has the confidentiality assurance of the TVP y, but A could not deduce the freshness of the TVP y.

4) At the end of the protocol run, A has computed the key "$k = \alpha^{bx+ay} \bmod p$". From Lemma 4.2, Lemma 4.3, and Lemma 4.4, A has gotten the confidentiality and freshness assurances of k from the confidentiality and freshness of the TVP x, and A has gotten the association of A and B with k from the TVP x and B's long-term key b respectively. Similar cases exist for B. At last, neither party obtains entity authentication, hence the adversary could launch an attack with a masquerade.

From Table 4.15 we get security properties of the protocol. The freshness assurance of the session key k depends on the fresh input x or y from each party, and the association of k depends on the long-term private key a and b. Neither party obtains entity authentication or key confirmation. The adversary can launch an attack by impersonating A directly.

Table 4.15 Security analysis of the MTI key agreement in two-pass

	A				B			
	B	x	y	k	A	x	y	k
Message 1		11#				1?#		
Message 2			1?#				11#	
End of run		11#	1?#	11AB		1?#	11#	11AB

Hence, the protocol in Example 4.6 may suffer an attack[29], as shown in Fig. 4.8.

$$\begin{aligned}
&\text{Message 1} \quad A \to I(B): \quad \alpha^x \bmod p\\
&\text{Message 1}' \quad I \to B: \quad \alpha^x \bmod p\\
&\text{Message 2}' \quad B \to I: \quad \alpha^y \bmod p\\
&\text{Message 2} \quad I(B) \to A: \quad \alpha^{ey} \bmod p
\end{aligned}$$

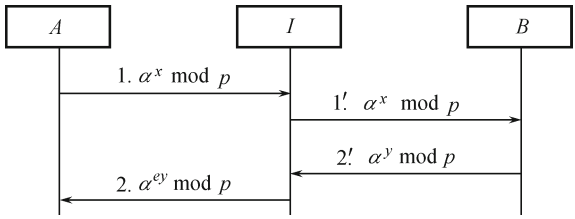

Fig. 4.8 The attack on the MTI key agreement protocol in two-pass.

Premise

A selects a random integer "a $(1 \leqslant a \leqslant p-2)$" as A's long-term private key, and registers the public-key "$\alpha^a \bmod p$"; B selects a random integer "b $(1 \leqslant b \leqslant p-2)$" as B's long-term private key, and registers the public-key "$\alpha^b \bmod p$"; the adversary I selects an integer e, computes "$\alpha^{ae} \bmod p$", and registers the public-key "$\alpha^{ae} \bmod p$".

Protocol actions

1) In Message 1, A randomly chooses a secret "x, $1 \leqslant x \leqslant p-2$", and sends B message "$\alpha^x \bmod p$".

2) In Message 1′, I launches a new protocol run between I and B by forwarding A′s exponential α^x to B.

3) B receives α^x and computes the shared key between I and B as "$k = (\alpha^x)^b(\alpha^{ae})^y = \alpha^{bx+aey} \bmod p$".

4) In Message 2′, I intercepts B's exponential α^y, then modifies "α^y to α^{ey}" and sends it to A.

5) A receives α^{ey} and computes the shared key with B as "$k = (\alpha^{ey})^a(\alpha^b)^x = \alpha^{aey+bx} \bmod p = \alpha^{bx+aey} \bmod p$".

At the end of the protocol run, A believes it is shared key "α^{bx+aey}" with B, while B believes it is shared key "α^{bx+aey}" with I. In this attack, I is not actually able to compute k itself, but rather causes B to have false beliefs. B concludes that subsequently received messages encrypted by the key "$k = \alpha^{bx+aey} \bmod p$" originated from I, whereas, in fact, it is only A who knows k and can originate such messages. This attack may be detected by key confirmation and prevented by modifying the protocol so that the exponentials or the identities of the intended entities are authenticated, e.g., through a digital signature.

4.3.3 Message replay attack

Message replay is a classic attack on authentication and authenticated key establishment protocols. A message replay attack is where a previous legitimate data transmission is captured or recorded and then replayed by an attacker in a new protocol run attempting to gain unauthorized access to data or re-

sources. A replay attack can be used in conjunction with a masquerade where an unauthorized user pretends to be somebody else.

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack by IP packet substitution (such as stream cipher attack).

Suppose A wants to prove A's identity to B. B requests A's password as proof of identity A dutifully provides (possibly after some transformation like a hash function); meanwhile, Malice is eavesdropping the conversation and keeps the password. After the interchange is over, Malice connects with B posing as A; when asking for a proof of identity, Malice sends A's password read from the last session which B will accept.

A way to avoid replay attacks is using session tokens: B sends a one-time token to A, which A uses to transform the password and sends the result to B (e.g., computing a hash function of the session token appended to the password). On B's side, B performs the same computation; if and only if both values match, the login is successful. Now suppose Malice has captured this value and tries to use it on another session; B sends a different session token, and when Malice replies with the captured value it will be different from B's computation. Session tokens should be chosen by a (pseudo-) random process. Otherwise Malice may be able to guess some future token and convince A to use that token in A's transformation. Malice can then replay A's reply at a later time, which B will accept.

B can also send nonces but should then include a message authentication code (MAC), which A should check in order to avoid replay attacks.

Timestamping is another way of preventing a replay attack. Synchronization should be achieved when using timestamp in a secure protocol. For example, B periodically broadcasts the time on B's clock together with a MAC. When A wants to send B a message, A includes A's best estimate of the time on A's clock in A's message, which is also authenticated. B only accepts messages for which the timestamp is within a reasonable tolerance. The advantage of this scheme is that B does not need to generate (pseudo-) random numbers.

It seems that we have already established a good awareness of message-replay attacks. This can be evidently seen from the ubiquitous use of TVPs (nonces, timestamps) in the basic and standard protocol constructions. However, simply using a timestamp on data or a message, or using tokens to verify timestamps of messages does not guarantee the key freshness. In essence, replay attack is generally applicable to a communication protocol where key freshness assurance is absent. This is why mistakes can be made repeatedly even when the designers know the errors very well in a different context. The freshness assurance could be achieved as in Lemma 4.3.

Example 4.7 Recall Needham-Schroeder shared key protocol in Example

3.17. The protocol as shown in Fig. 4.9 intends to establish a new session key k_{AB} between A and B, with the help of the trusted server S. N_A, N_B are nonces generated by A and B respectively; K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively.

$$
\begin{aligned}
&\text{Message 1} \quad A \rightarrow S : A, B, N_A \\
&\text{Message 2} \quad S \rightarrow A : \{N_A, k_{AB}, B, \{k_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&\text{Message 3} \quad A \rightarrow B : S, \{k_{AB}, A\}_{K_{BS}} \\
&\text{Message 4} \quad B \rightarrow A : \{N_B\}_{k_{AB}} \\
&\text{Message 5} \quad A \rightarrow B : \{N_B - 1\}_{k_{AB}}
\end{aligned}
$$

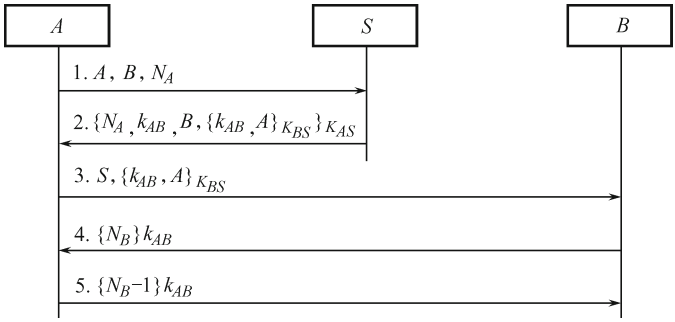

Fig. 4.9 The Needham-Schroeder shared key protocol.

Protocol security analysis

1) In Message 1, from Lemma 4.3, A has the freshness assurance of the TVP N_A.

2) Upon receiving Message 2, A has the assurance that Message 2 including trusted freshness N_A must be encrypted by the trusted third party S, and could not be a replay one. From Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the new chosen session key k_{AB}; from Lemma 4.4, A has the association assurance of the new session key k_{AB} with A and B since Message 2 could not be a replay one, and only S could encrypt N_A and k_{AB}. From Lemma 4.1, A has gotten the entity authentication of S.

3) Upon receiving Message 3, from Lemma 4.2, B has the assurance that the new chosen session key k_{AB} is confidential, but B does not know whether k_{AB} is a new generated key for this protocol run or a promised one, and B does not know whether k_{AB} is associated with A and B in this protocol run.

4) Upon receiving Message 4, A decrypts N'_B via using k_{AB}, but A is not sure whether N'_B is exactly the randomly chosen TVP N_B by B, or just a value decrypted using k_{AB} from a random data selected by the adversary I.

5) Upon receiving Message 5, B could not get any new assurance about the protocol security.

From Table 4.16, the absence of the security properties, various attacks, most of which are message replay attacks, could be constructed.

Table 4.16 Security analysis of the Needham-Schroeder shared key protocol

	A					B				
	B	S	N_A	N_B	k_{AB}	A	S	N_A	N_B	k_{AB}
Message 1			01#					0?#		
Message 2		1			11AB					
Message 3										1?#
Message 4				1?#					11#	
Message 5										
End of run		1	01#	1?#	11AB			0?#	11#	1?#

1. Attack on Needham-Schroeder shared key protocol by impersonating B

Example 4.8 From the absence of the B's liveness, in the point of view of A, an attack by impersonation B could be launched, as shown in Fig. 4.10.

$$\begin{aligned}
&\text{Message 1} \quad A \to S: \quad A, B, N_A \\
&\text{Message 2} \quad S \to A: \quad \{N_A, k_{AB}, B, \{k_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&\text{Message 3} \quad A \to I(B): S, \{k_{AB}, A\}_{K_{BS}} \\
&\text{Message 4} \quad I(B) \to A: N \\
&\text{Message 5} \quad A \to I(B): \{N'_B - 1\}_{k_{AB}}, \text{ where } N'_B = \{N\}_{k_{AB}^{-1}}
\end{aligned}$$

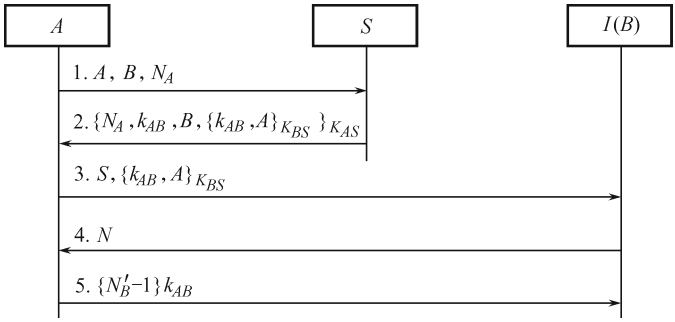

Fig. 4.10 An attack on the Needham-Schroeder shared key protocol by impersonating B.

Protocol actions

1) The message exchanges from Message 1 to Message 3 are the same as in the original Needham-Schroeder shared key protocol.

2) In Message 4, B sends a random nonce N to A as $\{N_B\}_{k_{AB}}$, then A decrypts N using the new session key k_{AB} and gets $N'_B = \{N\}_{k_{AB}^{-1}}$.

3) In Message 5, A encrypts $\{N'_B - 1\}$ using k_{AB} as a response to B's

challenge N'_B (Actually N'_B is not a challenge from B, it is only a nonce from the attacker).

Upon termination of the protocol run in Example 4.8, the adversary I is not actually able to get k_{AB} itself, but rather causes A to have false beliefs: A has completed a successful protocol run with B, and is sharing a new session key k_{AB} with B. A concludes that subsequently messages could be encrypted using k_{AB} and safely transmitted to B, whereas in fact, B knows nothing about the key establishment procedure.

2. An attack on the Needham-Schroeder shared key protocol by using compromised key

Example 4.9 From the absence of the key freshness assurance of k_{AB}, the Needham-Schroeder protocol is vulnerable to an attack discovered by Denning and Sacco[31], as illustrated in Fig. 4.11. The attacker I intercepts A's messages sent by and to A in the message lines 3, 4 and 5, and replays old session key material $S, \{k'_{AB}, A\}_{K_{BS}}$ which the attacker may record from a previous run of the protocol between A and B. An old session key k'_{AB} could possibly be promised since a careless communication principal may put it in an insecure place, or discard it etc., while the attacker has unlimited time to spend on finding an old data encryption key and then reusing it as though it were new.

$$
\begin{array}{ll}
\text{Message 1} & A \rightarrow S: \quad A, B, N_A \\
\text{Message 2} & S \rightarrow A: \quad \{N_A, k_{AB}, B, \{k_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
\text{Message 3} & A \rightarrow B: \quad S, \{k_{AB}, A\}_{K_{BS}} \\
\text{Message 3}' & I(A) \rightarrow B: S, \{k'_{AB}, A\}_{K_{BS}} \\
\text{Message 4}' & B \rightarrow I(A): \{N_B\}_{k'_{AB}} \\
\text{Message 5}' & I(A) \rightarrow B: \{N_B - 1\}_{k'_{AB}}
\end{array}
$$

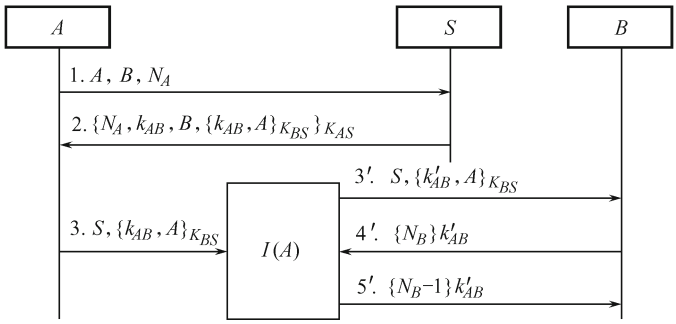

Fig. 4.11 An attack on the Needham-Schroeder shared key protocol using a compromised key k'_{AB}.

Protocol actions

1) The message exchanges from Message 1 to Message 3 are the same as in the original Needham-Schroeder shared key protocol.

2) In Message $3'$, the attacker I sends a recorded old message "$S, \{k'_{AB}, A\}_{K_{BS}}$" to B where I has the knowledge of the old session key k'_{AB}.

3) In Message $4'$, B sends A a challenge N_B to confirm the new session key.

4) In Message $5'$, A makes response $\{N_B - 1\}_{k'_{AB}}$ to B by using a compromised old session key k'_{AB}.

Upon termination of the protocol run in Example 4.9, A believes that the key establishment with B fails, whereas, B believes that he has successfully established a new session key k'_{AB} with A, and B may ignore the subsequently key establishment requirement for a new session key. Actually, k'_{AB} is a promised key known by the attacker. In deed, this attack could be launched by I from sending Message $3'$ directly, and the principal A will not participate in this protocol run at all (Example 4.10).

3. Attack on Needham-Schroeder shared key protocol by impersonating A

Example 4.10 From the absence of the A's liveness, the attacker may launch an attack without the presence of A, as shown in Fig. 4.12.

$$\begin{aligned}
&\text{Message } 3' \quad I(A) \rightarrow B : S, \{k'_{AB}, A\}_{K_{BS}} \\
&\text{Message } 4' \quad B \rightarrow I(A) : \{N_B\}_{k'_{AB}} \\
&\text{Message } 5' \quad I(A) \rightarrow B : \{N_B - 1\}_{k'_{AB}}
\end{aligned}$$

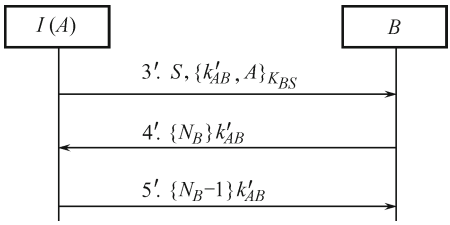

Fig. 4.12 An attack on the Needham-Schroeder shared key protocol by impersonating A.

4.3.4 Parallel session attack

A parallel session attack occurs when two or more protocol runs are executed concurrently and messages from one run (the reference session) are used to form spoofed messages in another run (the attack session). Following are examples of the parallel session attack.

Example 4.11 Figure 4.13 is a simple one-way authentication protocol. A wants to check the liveness of B by using a new chosen challenge N_A.

$$\begin{array}{ll} \text{Message 1} & A \rightarrow B : \{N_A\}_{K_{AB}} \\ \text{Message 2} & B \rightarrow A : \{N_A + 1\}_{K_{AB}} \end{array}$$

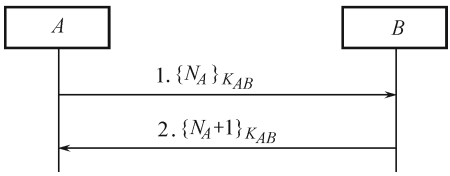

Fig. 4.13 A simple one-way authentication protocol.

Notation

A and B are two protocol principals.

Premise

K_{AB} is the shared long-term key between A and B, which is initially established by non-cryptographic, and out-of-band techniques; N_A is a nonce randomly chosen by A.

Protocol actions

1) In Message 1, A randomly chooses a new nonce N_A as a challenge for this protocol run, and sends it to B encrypted under the shared long-term key K_{AB} between A and B.

2) Upon receiving Message 1, B gets N_A from the encryption $\{N_A\}_{K_{AB}}$, and responses $\{N_A + 1\}_{K_{AB}}$ to show that B is operational.

Successful execution of the protocol should convince A that B is present since only B could have formed the appropriate response $\{N_A + 1\}_{K_{AB}}$ to the challenge N_A issued in Message 1.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and the freshness assurances of the TVP N_A.

2) Upon receiving Message 1, B could not determine whether $\{N_A\}_{K_{AB}}$ is a nonce chosen by the attacker or a challenge selected by the opponent party A.

3) Upon receiving Message 2, A could not determine whether $\{N_A+1\}_{K_{AB}}$ is an appropriate response to the challenge N_A from B or not, since there does not exist the evidence that $\{N_A + 1\}_{K_{AB}}$ is accomplished by the principal B, hence it even may be a trapped one-way transformation from A itself. Table 4.17 indicates the analyzing result.

Table 4.17 Security analysis of the simple one-way authentication protocol

	A	
	B	N_A
Message 1		11#
Message 2		
End of run		11#

Example 4.12 Figure 4.14 illustrates that an intruder can play the role of B as responder and initiator. The attack works by starting another protocol run in response to the initial challenge $\{N_A\}$.

$$\begin{aligned}
&\text{Message 1} \quad A \rightarrow I(B) : \{N_A\}_{K_{AB}} \\
&\text{Message 1}' \quad I(B) \rightarrow A : \{N_A\}_{K_{AB}} \\
&\text{Message 2}' \quad A \rightarrow I(B) : \{N_A + 1\}_{K_{AB}} \\
&\text{Message 2} \quad I(B) \rightarrow A : \{N_A + 1\}_{K_{AB}}
\end{aligned}$$

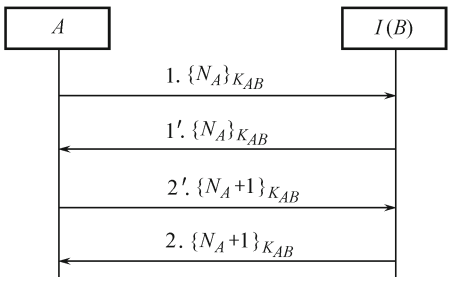

Fig. 4.14 A parallel session attack on the simple one-way authentication protocol.

Notation

A and B are two protocol principals, I is the attacker.

Premise

K_{AB} is the shared long-term key between A and B, which is initially established by non-cryptographic, and out-of-band techniques; N_A is a nonce randomly chosen by A.

Protocol actions

1) To initiate the attack, the adversary waits for A to initiate the first protocol session with B. A does the same thing as in Message 1 of Example 4.11.

2) I intercepts the Message 1 and pretends to be B, starting a second run of the protocol by replaying the intercepted message $\{N_A\}_{K_{AB}}$.

3) A replies to $I(B)$'s challenge in Message 2′ with the exact value $\{N_A + 1\}_{K_{AB}}$ that $I(B)$ requires to accurately complete the attack session.

4) I intercepts the Message $2'$ and replays the intercepted message $\{N_A + 1\}_{K_{AB}}$ to A as B's (indeed, it is I) response to Message 1.

Successful execution of the attack on this simple One-Way protocol could convince A that B is present since only B could have formed the appropriate response $\{N_A + 1\}_{K_{AB}}$ to the challenge N_A issued in Message 1.

Such attacks may be prevented via modifying the protocol so that the challenge response messages could show the identity of the party performing encryption or decryption. For example, change $\{N_A + 1\}_{K_{AB}}$ to $\{A, N_A + 1\}_{K_{AB}}$, or $\{B, N_A + 1\}_{K_{AB}}$.

Example 4.13 The Woo-Lam protocol[5] is an authentication protocol based on symmetric-key cryptography, as shown in Fig. 4.15. The protocol intends to authenticate A to B with the aid of a trusted third party S. The nonce N_B servers as challenge for authenticating A to B.

$$
\begin{array}{ll}
\text{Message 1} & A \rightarrow B : A \\
\text{Message 2} & B \rightarrow A : N_B \\
\text{Message 3} & A \rightarrow B : \{N_B\}_{K_{AS}} \\
\text{Message 4} & B \rightarrow S : \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
\text{Message 5} & S \rightarrow B : \{N_B\}_{K_{BS}}
\end{array}
$$

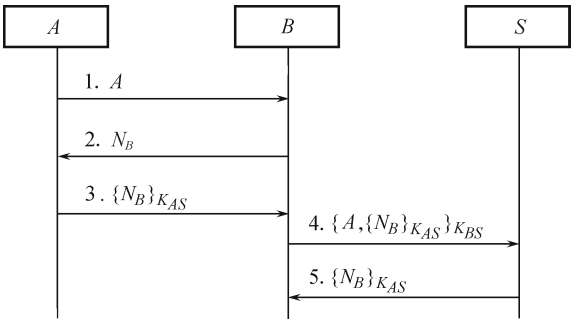

Fig. 4.15 The Woo-Lam authentication protocol.

Notation

A and B are two protocol principals, and S is the trusted third party. K_{AS} and K_{BS} are keys that A and B shared with S respectively.

Premise

K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively, which are initially established by non-cryptographic, and out-of-band techniques. N_B is a nonce randomly chosen by B.

Protocol actions

1) In Message 1, A launches a new protocol run.

2) In Message 2, B randomly chooses a new nonce N_B as a challenge to this protocol run and sends it to A.

3) In Message 3, A sends B a response to the challenge N_B using the shared long-term only known by A and the trusted third party S, to show that it is A who has encrypted B's challenge N_B.

4) In Message 4, B encrypts A's response with A's identity using the shared long-term key only known by B and the trusted third party S, to show that it is B who has encrypted A's response $\{N_B\}_{K_{AS}}$.

5) Upon receiving Message 4, S checks A's response $\{N_B\}_{K_{AS}}$ to confirm A's identity. Then, S sends B the message $\{N_B\}_{K_{BS}}$ showing that A's identity has been authenticated by S.

6) Upon receiving Message 5, B checks S's response $\{N_B\}_{K_{BS}}$ to get N_B. If N_B is correct, then B believes that A's identity has been authenticated by the trusted third party S.

Successful execution should convince B that A is present with the help of the trusted party S.

Protocol security analysis

1) In Message 1, neither A nor B can draw any useful assurance from it.

2) In Message 2, from Lemma 4.3, B has the freshness assurance of the TVP N_B.

3) Upon receiving Message 3, B could not determine whether $\{N_B\}_{K_{AS}}$ is a nonce chosen by the attacker or a response from the opponent party A, and B can only treat it as an unrecognizable foreign cipher chunk. Neither A nor B can draw any new assurance from Message 3.

4) Upon receiving Message 4, S could not determine whether $\{N_B\}_{K_{AS}}$ or $\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$ is a fresh message or not since S doesn't have any trusted freshness. Actually, S could not authenticate the liveness of A and B, but S proves that N_B is recovered from an encryption under the shared long-term key between A and S.

5) Upon receiving Message 5, from Lemma 4.1, B believes that it must be S who has decrypted the message $\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$ and $\{N_B\}_{K_{AS}}$ from recovering N_B which is trusted by B. However, B cannot authenticate the liveness of A since there does not exist any evidence that N_B is the challenge from B to A.

Upon termination of the Woo-Lam protocol, B has not gotten the assurance that A is present. The analyzing result is indicated in Table 4.18.

From the absence of the liveness property of A, an attacker can play the role of A as responder or initiator. In addition to the attacks discovered in [10] and [12] on the Woo-Lam Protocol, a new attack is illustrated in Example 3.13. In essence, a parallel session attack continually exists in the Woo-Lam Protocol, since the absence of A's liveness property has not been fixed.

Table 4.18 Security analysis of the Woo-Lam protocol

	A			B			S		
	B	S	N_B	A	S	N_B	A	B	N_B
Message 1									
Message 2			0?#			01#			0?#
Message 3									
Message 4						01#			0?#
Message 5					1				
End of run					1				0?#

4.3.5 Reflection attack

Reflection attack is a type of replay attack in which transmitted data is sent back to its originator. In a basic authentication scheme, a secret is known to both the originator and the target (or the trusted server), this allows them to be authenticated and they may verify this shared secret without sending it in plaintext over the wire. The originator initiates a connection to a target, and the target attempts to authenticate the originator by sending it a challenge, then the originator utilizes the shared secret to process this randomly chosen challenge to show his identity. The essential idea of the reflection attack is to trick the target into providing the answer to its own challenge (Example 4.11). That is, the same challenge-response protocol is used by each side to authenticate the other side, thereby leaving the attacker with fully-authenticated channel connection.

Example 4.11 could be fixed by sending the responder's identity within the response. Then, if the originator receives a response that has its own identity in it, then the originator will reject the response; if the originator receives a response that has the opponent's identity in it, and if the nonce is the same as the one the originator has sent in his challenge, then the originator will accept the message.

Example 4.14 Recall the fixed Woo-Lam protocol by Abadi and Needham[10]. This fixed version of the Woo-Lam Protocol suffers a reflection attack discovered by Clark and Jacob[12]. Here, the attacker mounts reflection attack twice: Message 3 is a reflection of Message 2, and Message 5 is that of Message 4. First, the random chunk that B receives in Message 3 is actually B's nonce sent out in Message 2. Again, the cipher chunk that B receives in Message 5 is actually the one created by himself and sent out in Message 4. B cannot detect this attack discovered by Clark and Jacob.

As Mao has stated in [22], a series of fixes for the Woo-Lam Protocol[32] are also flawed in a similar way: they all suffer reflection attack in various ways[10, 12, 22]. The key reason for this flaw is the absence of the liveness

property of A as we have illustrated in Table 4.18.

4.3.6 Interleaving attack

Interleaving attack is a type of replay attack in which transmitted data is from outside the current run of the protocol. The attacker may compose a message and sends it out to a principal in one run, from which he expects to receive an answer; the answer may be useful for another principal in another run, and in the latter run, the answer obtained from the former run may further stimulate the latter principal to answer a question which in turn is further used in the first run, and so on[22]. In essence, interleaving attack is generally applicable to a communication protocol where a principal's liveness or the session key's association assurance is absent. Here are two examples:

Example 4.15 Figure 4.16 illustrates a refined Woo-Lam protocol in [22]. The absence of the A's liveness has not been solved yet (see Table 4.18).

$$
\begin{aligned}
&\text{Message 1} \quad A \rightarrow B : A \\
&\text{Message 2} \quad B \rightarrow A : N_B \\
&\text{Message 3} \quad A \rightarrow B : \{N_B\}_{K_{AS}} \\
&\text{Message 4} \quad B \rightarrow S : \{A, N_B, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
&\text{Message 5} \quad S \rightarrow B : \{N_B\}_{K_{BS}}
\end{aligned}
$$

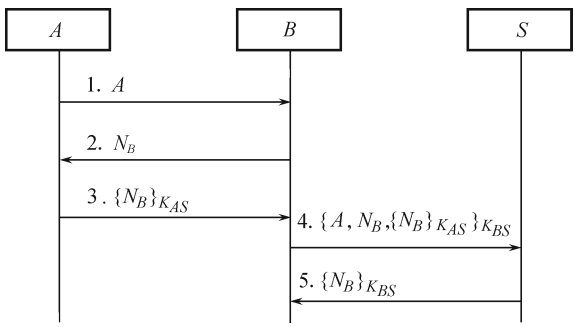

Fig. 4.16 The refined Woo-Lam authentication protocol version of Mao.

Protocol actions

1) The message exchanges from Message 1 to Message 3 are the same as in the original Woo-Lam authentication protocol.

2) In Message 4, $\{N_B\}_{K_{AS}}$ is an encryption which B could not recover, so B includes the trusted freshness N_B in Message 4 to guarantee the freshness of Message 4.

3) Upon receiving Message 4, S recovers N_B using K_{BS}, checks whether A has responded to the same challenge N_B, and then re-encrypts N_B using the long-term key K_{BS} and sends N_B back to B in Message 5.

4) Upon receiving Message 5, If S replies to $\{N_B\}_{K_{BS}}$, then B is convinced that A is active in this protocol run.

Successful execution should convince B that A is present with the help of the trusted party S.

Protocol security analysis

1) The assurances from Message 1 to Message 3 are the same as in the original Woo-Lam authentication protocol.

2) Upon receiving Message 4, since there is none trusted TVP for S, S still could not determine whether $\{N_B\}_{K_{AS}}$ or $\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$ is a fresh message or not. Actually, S still could not authenticate the liveness of A and B.

Hence, upon termination of the Woo-Lam protocol, B has not gotten the assurance that A is present. The security properties of the Woo-Lam protocol are the same as those in Table 4.18. Hence there exists an interleaving attack on the refined Woo-Lam protocol, as shown in Fig. 4.17.

$$
\begin{array}{ll}
\text{Message } 1 & A \rightarrow I : A \\
\text{Message } 1' & I(A) \rightarrow B : A \\
\text{Message } 2' & B \rightarrow I(A) : N_B \\
\text{Message } 2 & I \rightarrow A : N_B \\
\text{Message } 3 & A \rightarrow I : \{N_B\}_{K_{AS}} \\
\text{Message } 3' & I(A) \rightarrow B : \{N_B\}_{K_{AS}} \\
\text{Message } 4' & B \rightarrow S : \{A, N_B, \{N_B\}_{K_{AS}}\}_{K_{BS}} \\
\text{Message } 5' & S \rightarrow B : \{N_B\}_{K_{BS}} \\
\text{Message } 4 & I \rightarrow S : \{I, N_B, \{N_B\}_{K_{AS}}\}_{K_{IS}} \\
\text{Message } 5 & S \rightarrow I : \{N_B\}_{K_{IS}}
\end{array}
$$

Protocol actions

1) In Message 1, A tells I that A wants to establish a connection with I; upon receiving Message 1, I establishes a connection with B instantly by impersonating A.

2) In Message $2'$, B provides challenge N_B to the session between A (indeed, it is I) and B; upon receiving Message $2'$, I provides the same challenge N_B to A for the session between A and I.

3) In Message 3, A returns this challenge N_B encrypted under K_{AS} to I; upon receiving Message 3, by impersonating A, I passes $\{N_B\}_{K_{AS}}$ to B as A's response to the challenge N_B for the session between A and B. As we have seen, in Message 3, A servers as an encryption oracle in the session between $I(A)$ and B.

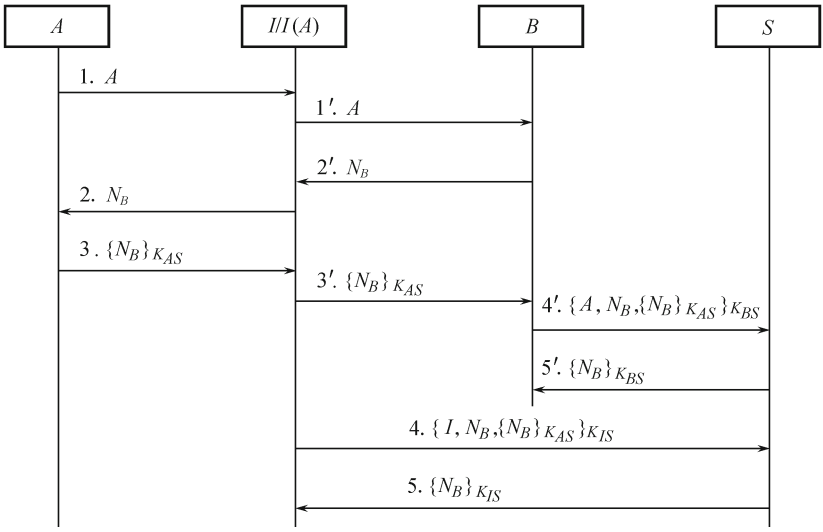

Fig. 4.17 An attack on the refined Woo-Lam protocol version of Mao.

4) Upon receiving Message 3′, B passes the encryption $\{N_B\}_{K_{AS}}$ on to S in Message 4′ for future verification.

5) Upon receiving Message 4′, S could not find any abnormality since the received message is an encryption under K_{BS} and K_{AS}.

6) Message 5′ is the reply from S which contains the challenge N_B intended for A and B. On the basis of the reply containing N_B, B believes that A is active in this protocol run.

7) Upon receiving Message 3 containing $\{N_B\}_{K_{AS}}$, I can continue his session with A, and at last, successfully complete the protocol run.

This is a perfect attack, all principals including A, B and S could not find any abnormality. Upon termination of the run of this refined Woo-Lam protocol, B accepts "the run with A", but in fact, A has not launched the run with B at all, and A thinks that A has completed a protocol run with I.

As we have shown above, although the principal name A is explicitly mentioned in Message 4, the absence of A's liveness still causes this flaw.

Example 4.16 Recall the Needham-Schroeder public-key authentication protocol and the security analysis of the Needham-Schroeder public-key protocol in Table 4.12. Since B could not guarantee the freshness of N_A and the association of N_A with A and B, there exists the interleaving attack discovered by Lowe[9].

Example 4.17 Recall Mao's revised Needham-Schroeder public-key 0-* protocol as shown in Fig. 4.18.

$$\text{Message 1} \quad A \to B : \left\{\{A, N_A\}_{K_A^{-1}}\right\}_{K_B}$$

Message 2 $B \rightarrow A : \left\{N_A, \{N_B\}_{K_B^{-1}}\right\}_{K_A}$

Message 3 $A \rightarrow B : \left\{\{N_B\}_{K_A^{-1}}\right\}_{K_B}$

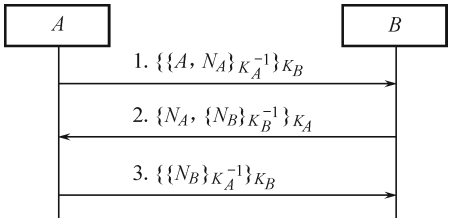

Fig. 4.18 The revised Needham-Schroeder public-key 0-* protocol.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and the freshness assurances of the TVP N_A.

2) Upon receiving Message 1, B could not determine whether $\{\{A, N_A\}_{K_A^{-1}}\}_{K_B}$ is a replay message from the attacker or a new message generated by the opponent party A. B could not get any security assurances.

3) In Message 2, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and the freshness assurances of the TVP N_B.

4) Upon receiving Message 2, according to the freshness assurance of N_A, A is sure that only B could get N_A, using B's private key K_B^{-1}, and send N_A back to A in Message 2, hence, from Lemma 4.1, A has the liveness assurance of B. Since only A could decrypt $\{N_A, \{N_B\}_{K_B^{-1}}\}_{K_A}$ to get N_B, and N_B is signed by B's private key K_B^{-1}, from Lemma 4.3 and Lemma 4.4, A has the freshness assurance and the association assurance of the TVP N_B with A and B. So does N_A.

5) Similar case exists as in Message 3: upon receiving Message 3, from Lemma 4.2, Lemma 4.3, and Lemma 4.4, B has the confidentiality assurance, the freshness assurance, and the association assurance of the TVP N_A and N_B with A and B; from Lemma 4.1, B has the liveness assurance of A.

Upon termination of the Needham-Schroeder public-key 0-* protocol run, both A and B achieve the security objects of the Needham-Schroeder public-key. The analyzing result is indicated in Table 4.19.

Table 4.19 Security analysis of the refined Needham-Schroeder public-key 0-* protocol

	A			B		
	B	N_A	N_B	A	N_A	N_B
Message 1		11A			1?#	
Message 2	1	11AB	11AB			11#
Message 3				1	11AB	11AB
End of run	1	11AB	11AB	1	11AB	11AB

Example 4.18 Recall Mao's another revised Needham-Schroeder public-key 1-* protocol as shown in Fig. 4.19.

$$\begin{aligned}
&\text{Message 1} \quad A \rightarrow B : \{A, \{N_A\}_{K_B}\}_{K_A^{-1}} \\
&\text{Message 2} \quad B \rightarrow A : \{\{N_A, N_B\}_{K_A}\}_{K_B^{-1}} \\
&\text{Message 3} \quad A \rightarrow B : \{\{N_B\}_{K_B}\}_{K_A^{-1}}
\end{aligned}$$

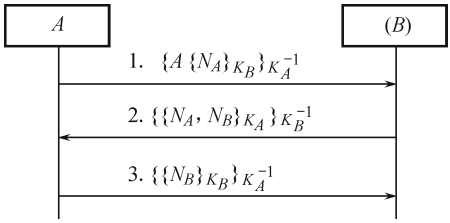

Fig. 4.19 The revised Needham-Schroeder public-key 1-* protocol.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and the freshness assurances of the TVP N_A.

2) Upon receiving Message 1, B could not determine whether $\{A, \{N_A\}_{K_B}\}_{K_A^{-1}}$ is a replay message from the attacker or a new message generated by the opponent party A. B could not get any security assurances.

3) In Message 2, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and the freshness assurances of the TVP N_B.

4) Upon receiving Message 2, according to the freshness assurance of N_A, A is sure that only B could get N_A, using B's private key K_B^{-1}, and send N_A back to A in Message 2. Hence, from Lemma 4.3 and Lemma 4.4, A has the freshness assurance and the association assurance of the TVP N_A and N_B with A and B; from Lemma 4.1, A has the liveness assurance of B.

5) Similar case exists in Message 3: upon receiving Message 3, from Lemma 4.2, Lemma 4.3, and Lemma 4.4, B has the confidentiality assurance, the freshness assurance, and the association assurance of the TVP N_A and N_B with A and B; from Lemma 4.1, B has the liveness assurance of A.

Upon termination of the Needham-Schroeder public-key 1-* protocol run, both A and B achieve the security objects of the Needham-Schroeder public-key. The analyzing result is indicated in Table 4.20.

Table 4.20 Security analysis of the refined Needham-Schroeder public-key 1-* protocol

	A			B		
	B	N_A	N_B	A	N_A	N_B
Message 1		11#			1?#	
Message 2	1	11AB	11AB			11#
Message 3				1	11AB	11AB
End of run	1	11AB	11AB	1	11AB	11AB

4.3.7 Attack due to type flaw

An attack is stated due to type flaw in [22]: typical type flaws include a principal tricked to misinterpret a nonce, a timestamp or an identity into a key, etc. Misinterpretations are likely to occur when a protocol is poorly designed in that the type information of message components is not explicit, then type flaws can be very common in implementation. Let us check why type flaws exist using the security analysis apporach based on trusted freshness. In essence, type flaw attacks are generally applicable to a communication protocol where entity authentication or the freshness assurance of a TVP is absent.

Example 4.19 Neuman and Stubblebine[33] propose an authentication protocol, as shown in Fig. 4.20, to achieve mutual authentication and authenticated key establishment between A and B with the help of a trusted third party S.

Message 1 $A \rightarrow B : A, N_A$
Message 2 $B \rightarrow A : B, \{A, N_A, T_B\}_{K_{BS}}, N_B$
Message 3 $A \rightarrow B : \{B, N_A, k_{AB}, T_B\}_{K_{AS}}, \{A, k_{AB}, T_B\}_{K_{BS}}, N_B$
Message 4 $A \rightarrow B : \{A, k_{AB}, T_B\}_{K_{BS}}, \{N_B\}_{k_{AB}}$

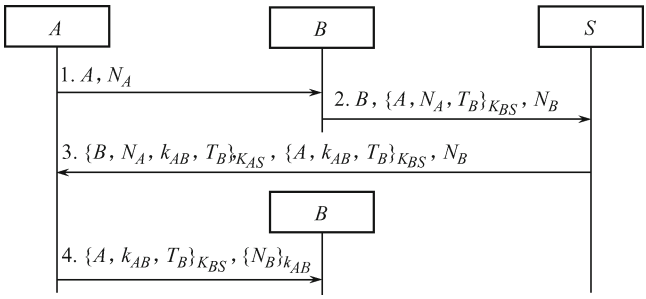

Fig. 4.20 The Neuman-Stubblebine authentication protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. N_A and N_B are nonces, and T_B is a timestamp generated by B referring to an absolute time. K_{AS} and K_{BS} are shared long-term keys, and k_{AB} is a new session key between A and B to be established in this authentication protocol.

Premise

K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively, which are initially established by non-cryptographic, and out-of-band techniques. N_A or N_B is a nonce randomly chosen by A and B respectively, and T_B is a timestamp generated by B.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identity A and a randomly chosen nonce N_A.

2) In Message 2, B randomly chooses a nonce N_B, and then sends the identity B, and the nonce N_B with the encryption $\{A, N_A, T_B\}_{K_{BS}}$ to S to indicate a new protocol run between A and B.

3) Upon receiving Message 2, S gets N_A and N_B for this run between A and B, and then S randomly chooses a new session key k_{AB} for this run and keeps it secret via an encryption under K_{AS} and K_{BS} respectively.

4) Upon receiving Message 3, A gets the new session key k_{AB} via the shared long-term key K_{AS} known only by A and S.

5) In Message 4, A forwards $\{A, k_{AB}, T_B\}_{K_{BS}}$ received in Message 3 to B, and encrypts N_B under k_{AB} to show A's knowledge of k_{AB} to B.

6) Upon receiving Message 4, B gets the new session key k_{AB} via the shared long-term key K_{BS} known only by B and S, and confirms A's knowledge of k_{AB} via the encryption $\{N_B\}_{k_{AB}}$.

Successful execution should convince A and B that both entities are present and k_{AB} is a new session key between A and B.

Unfortunately, this protocol has not achieved these security objects as it intends to.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A believes that the TVP N_A is no longer confidential, but it is fresh for this run.

2) Upon receiving Message 1, B could not get any assurance about this protocol.

3) In Message 2, from Lemma 4.2 and Lemma 4.3, B believes that the TVP N_B is no longer confidential, but it is fresh for this run.

4) Upon receiving Message 2, since T_B is an absolute timestamp, S believes that the message $\{A, N_A, T_B\}_{K_{BS}}$ is fresh, and it could not be a replay one. Furthermore, since K_{BS} is only known by B and S, from Lemma 4.1 and Lemma 4.3, S believes that B is present and it must be B who has just generated this message $\{A, N_A, T_B\}_{K_{BS}}$. From Lemma 4.4, S believes that

N_A and N_B are new established for the protocol run between A and B from the explicitly identity of A and the possession of the key K_{BS} by B.

5) Upon receiving Message 3, from the point of view of A, since N_A is a trusted fresh TVP and K_{AS} is only known by A and S, from Lemma 4.2 and Lemma 4.3, A gets the confidentiality and freshness assurances of the new session key k_{AB}. From Lemma 4.4, A believes that k_{AB} is for this protocol run between A and B from the explicitly identity of B and the possession of the key K_{AS} by A. According to the protocol semantic cue, the principal in $\{B, N_A, k_{AB}, T_B\}_{K_{AS}}$ is the one who has confirmed his identity and liveness to the third trusted party S. From Lemma 4.1, since N_A is a trusted fresh TVP from the point of view of A, A believes that it could only be S who has generated and sent the message $\{B, N_A, k_{AB}, T_B\}_{K_{AS}}$ after S has checked the liveness of B via the message of $\{A, N_A, T_B\}_{K_{BS}}$, hence A believes the liveness of S and B.

6) Upon receiving Message 4, since T_B is an absolute timestamp, B believes that the message $\{A, k_{AB}, T_B\}_{K_{BS}}$ is fresh, but the maximal term $\{A, k_{AB}, T_B\}_{K_{BS}}$ in Message 4 is similar to the maximal term $\{A, N_A, T_B\}_{K_{BS}}$ in Message 2, so $\{A, k_{AB}, T_B\}_{K_{BS}}$ may be a replay one of Message 2. Hence, B could not get any new assurance about this protocol.

The maximal term $\{A, k_{AB}, T_B\}_{K_{BS}}$ includes an identity of A, a random session key and an absolute timestamp (or we can call it a trusted freshness TVP), and it can be constructed via term definition, terms are T_B, $\{T_B\}_{K_{BS}}$, $\{k_{AB}, T_B\}_{K_{BS}}$, $\{A, T_B\}_{K_{BS}}$, then $\{A, k_{AB}, T_B\}_{K_{BS}}$. Similar case exists in the maximal term $\{A, N_A, T_B\}_{K_{BS}}$ of Message 2. Hence, $\{A, N_A, T_B\}_{K_{BS}}$ could be replayed in Message 4 as Mao has stated in [22]. Note that $\{N_B\}_{k_{AB}}$ is also a maximal term in Message 4.

Table 4.21 Security analysis of the Neuman-Stubblebine authentication protocol

	A					B				
	B	S	N_A	N_B	k_{AB}	A	S	N_A	N_B	k_{AB}
Message 1			01#					0?#		
Message 2									01#	
Message 3	1	1	01AB		11AB					
Message 4										
End of run	1	1	01AB		11AB			0?#	01#	

Upon termination of the protocol run, by analyzing of Table 4.21shows that B is not sure whether the opponent principal is present or not, and whether k_{AB} is a new session key for A and B or not.

From the absence of the liveness of the principal A and the association of k_{AB} with A and B in the point of view of B, the adversary I could launch an attack as shown in Fig. 4.21 by impersonating A[22].

$$\begin{array}{lll} \text{Message 1} & I(A) \rightarrow B: & A, N_A \\ \text{Message 2} & B \rightarrow I(S): & B, \{A, N_A, T_B\}_{K_{BS}}, N_B \end{array}$$

Message 3 $I(S) \rightarrow I(A)$: none

Message 4 $I(A) \rightarrow B$: $\{A, N_A, T_B\}_{K_{BS}}, \{N_B\}_{N_A}$

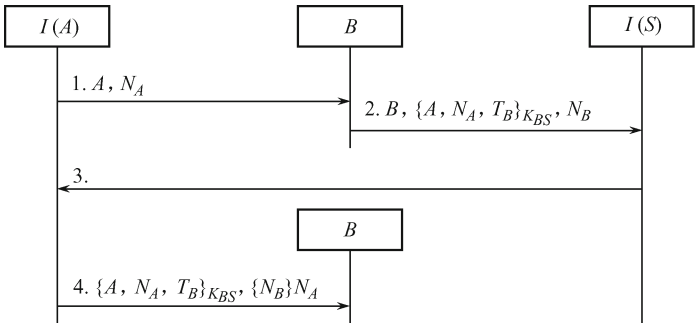

Fig. 4.21 An attack on the Neuman-Stubblebine authentication protocol.

Notation

$I(A)$ and $I(S)$ are adversaries impersonating A and S respectively.

Protocol actions

1) In Message 1, the adversary launches a new protocol run with a randomly chosen nonce N_A by impersonating A.

2) In Message 2, B does the same as in the original Neuman-Stubblebine authentication protocol.

3) I intercepts Message 2 to get N_B, then encrypts N_B using I's randomly chosen nonce N_A and sends $\{N_B\}_{N_A}$ to B to show A's knowledge of k_{AB} by confusing N_A with k_{AB}.

Upon termination of the protocol run, B believes that B has performed a successful protocol run with A, and N_A is a new session key for A and B, while A knows nothing about this key establishment procedure.

4.3.8 Attack due to name omission

Name omission is often the case in authentication protocols for the name information about a message can be deduced from other data parts in the context, or from what encryption keys have been applied to. To obtain an elegant protocol that contains little redundancy, protocol designers may omit the identities of the participants, which may lead to name-omission flaws. In security analysis based on trusted freshness, entity authentication or the association assurance of a TVP may be absent due to name omission. Hence, attack due to name omission can be constructed from these absences.

Example 4.20 Denning and Sacco propose a public-key protocol as an

alternative to their fix of the Needham-Schroeder shared key protocol[1, 31]. The protocol of Denning and Sacco is as shown in Fig. 4.22.

Message 1 $A \rightarrow S : A, B$

Message 2 $S \rightarrow A : \text{Cert}_A, \text{Cert}_B$

Message 3 $A \rightarrow B : \text{Cert}_A, \text{Cert}_B, \left\{\{k_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_B}$

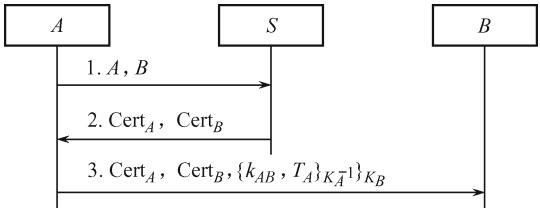

Fig. 4.22 The Denning and Sacco authentication protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. T_A is a timestamp generated by A referring to an absolute time. K_A (K_B) and K_A^{-1} (K_B^{-1}) are public-key and private key of A (B) respectively. Cert_A (Cert_B) is a certification of A's (or B's) identity and corresponding public-key (K_A) signed by a trusted certification authority center CA. k_{AB} is a new session key between A and B to be established in this authentication protocol.

Premise

Both A and B know the public-key of the trusted certification authority center CA to get K_A and K_B. Each principal knows the key pair of himself, that is, K_A and K_A^{-1} for A, and K_B and K_B^{-1} for B.

Protocol actions

1) In Message 1, A launches a new protocol run of mutual authentication and authenticated key establishment between A and B.

2) Upon receiving Message 2, A gets the public-key K_B of B.

3) In Message 3, $\{\{k_{AB}, T_A\}_{K_A^{-1}}\}_{K_B}$ is encrypted for confidentiality (under K_B) and authenticity (under K_A^{-1}).

4) Upon receiving Message 3, B gets the public-key K_A of A, and B sees that the new session key k_{AB} should be exclusively shared between A and B from B's private key K_B^{-1} and A's public-key K_A being applied. Note k_{AB} is randomly chosen by A for this protocol run.

Successful execution should convince A and B that k_{AB} is a new session key between A and B.

Unfortunately, the Denning and Sacco authentication protocol has not achieved the security objects as it intends to.

Protocol security analysis

1) In Message 1 and Message 2, neither A nor B could get any assurance about this protocol.

2) In Message 3, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and the freshness assurances of the new session key k_{AB}. From Lemma 4.4, A has the association assurance of k_{AB} with A, since the "loose" one-way transformation $\{k_{AB}, T_A\}_{K_A^{-1}}$ could only be generated by A.

3) Upon receiving Message 3, since T_B is an absolute timestamp, B believes that the message $\{k_{AB}, T_A\}_{K_A^{-1}}$ is fresh, and it could not be a replay one. Hence, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and the freshness assurances of the new session key k_{AB}. From Lemma 4.4, B has the association assurance of k_{AB} with A, since the "loose" one-way transformation $\{k_{AB}, T_A\}_{K_A^{-1}}$ could only be generated by A.

Table 4.22 means that upon termination of the protocol run B believes A is present, and the new session key k_{AB} is confidential, fresh, and associated with A but B.

Table 4.22 Security analysis of the Denning-Sacco protocol

	A		B	
	B	k_{AB}	A	k_{AB}
Message 1				
Message 2				
Message 3		11A	1	11A
End of run		11A		11A

Example 4.21 From the absence of the k_{AB}'s association with B in the point of view of B, the adversary can perform an attack by confusing the session key k'_{AB}, as shown in Fig. 4.23, intended for A and I to be the session key between A and B[10].

$$
\begin{aligned}
&\text{Message 1} \quad A \to S: && A, I\\
&\text{Message 2} \quad S \to A: && \mathrm{Cert}_A,\ \mathrm{Cert}_I\\
&\text{Message 3} \quad A \to I: && \mathrm{Cert}_A,\ \mathrm{Cert}_I,\ \left\{\{k'_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_I}\\
&\text{Message 1}' \quad I(A) \to S: && A, B\\
&\text{Message 2}' \quad S \to I(A): && \mathrm{Cert}_A,\ \mathrm{Cert}_B\\
&\text{Message 3}' \quad I(A) \to B: && \mathrm{Cert}_A,\ \mathrm{Cert}_B,\ \left\{\{k'_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_B}
\end{aligned}
$$

Notation

$I(A)$ is an adversary I impersonating A. K_I and K_I^{-1} are the public-key and private key of I. Cert_I is a certification of I's identity and corresponding public-key (K_I) signed by a trusted certification authority center CA.

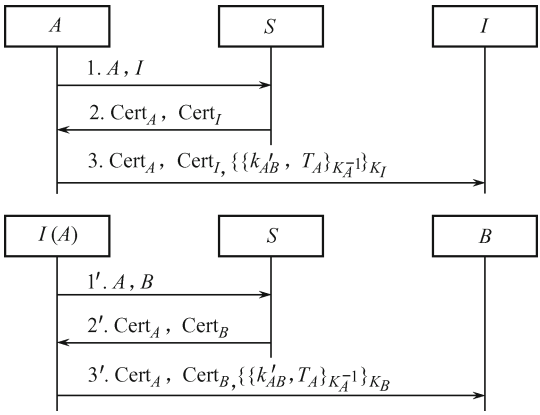

Fig. 4.23 An attack on the Denning-Sacco authentication protocol.

Premise

All principals A, B and I know the public-key of the trusted certification authority center CA to get K_A, K_B and K_I. Each principal knows the key pair of himself, that is, K_A and K_A^{-1} for A, K_B and K_B^{-1} for B, and K_I and K_I^{-1} for I.

Protocol actions

1) In Message 1, A launches a new protocol run of mutual authentication and makes authenticated key establishment between A and I.

2) Upon receiving Message 2, A gets the public-key K_I of I.

3) In Message 3, $\{\{k'_{AB}, T_A\}_{K_A^{-1}}\}_{K_I}$ is encrypted for confidentiality (under K_I) and authenticity (under K_A^{-1}).

4) Upon receiving Message 3, I gets the new session key k'_{AB}. Then I could launch an attack by impersonating A and use the transformation $\{k'_{AB}, T_A\}_{K_A^{-1}}$ to generate Message 3′ $\{\text{Cert}_A, \text{Cert}_B, \{\{k'_{AB}, T_A\}_{K_A^{-1}}\}_{K_B}\}$.

Upon termination of the protocol attack, B believes that B has performed a successful run with A, and k'_{AB} is a new session key for A and B, while A knows nothing about this key establishment between A and B, and k'_{AB} is actually shared by A and I.

4.3.9 Attack due to misuse of cryptographic services

The attack due to misuse of cryptographic services is a very common design flaw as stated in [22]. Misuse of cryptographic services means that a cryptographic algorithm used in a protocol provides an incorrect protection so that the needed protection is absent. This type of flaw may lead to various attacks.

Example 4.22 Recall Example 3.15, the Otway and Rees key establishment protocol is illustrated as in Fig. 4.24.

Message 1 $A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$
Message 2 $B \rightarrow A : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$
Message 3 $S \rightarrow B : M, \{N_A, k_{AB}\}_{K_{AS}}, \{N_B, k_{AB}\}_{K_{BS}}$
Message 4 $A \rightarrow B : M, \{N_A, k_{AB}\}_{K_{AS}}$

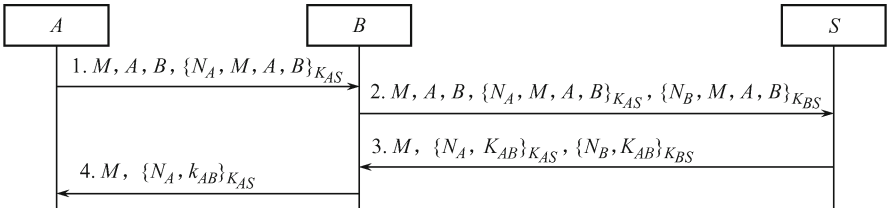

Fig. 4.24 The Otway-Rees key establishment protocol.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and the freshness assurances of the TVP N_A and the freshness assurance of the TVP M. From Lemma 4.4, A has the association assurance of N_A with A and B, since the transformation $\{N_A, M, A, B\}_{K_{AS}}$ generated by A includes the identities of both A and B, which implies the association of both A and B.

2) Upon receiving Message 1, B has the confidentiality assurance of the TVP N_A.

3) In Message 2, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and the freshness assurances of the TVP N_B. From Lemma 4.4, B has the association assurance of N_B with A and B, since the transformation $\{N_B, M, A, B\}_{K_{BS}}$ generated by B includes the identities of both A and B, which implies the association of both A and B.

4) Upon receiving Message 2, S could not get any assurance about this protocol. A could not get any new assurance since $\{N_B, M, A, B\}_{K_{BS}}$ is an encryption for B without corresponding decrypted key K_{BS}.

5) Upon receiving Message 3, from Lemma 4.3, B has the liveness assurance of the trusted third party S from the transformation $\{N_B, k_{AB}\}_{K_{BS}}$ including B's trusted freshness component N_B, since only S could get N_B and generate this message $\{N_B, k_{AB}\}_{K_{BS}}$. From the one-way transformation $\{N_B, k_{AB}\}_{K_{BS}}$ including the trusted freshness N_B, and from Lemmas 4.2, 4.3, and 4.4, B has the confidentiality assurance, freshness assurance of k_{AB}, and also the association assurance of k_{AB} with both A and B.

6) Similarly, upon receiving Message 4, from Lemma 4.2, Lemma 4.3, and Lemma 4.4, A has the liveness assurance of the trusted third party S from the receiving fresh nonce N_A, and A also has the confidentiality assurance,

freshness assurance of k_{AB}, and also the association assurance of k_{AB} with A and B. From Lemma 4.3, A has the liveness assurance of B from the receiving fresh nonce N_A and M.

Upon termination of this protocol run, B believes that S is alive from receiving fresh nonce N_B, and A believes that S is alive from receiving fresh nonce N_A, and B's liveness is also guaranteed to A via M, but A's liveness is not guaranteed to B in this protocol. The analyzing result is indicated in Table 4.23.

Table 4.23 Security analysis of the Otway-Rees protocol

	A					B				
	B	S	N_A	N_B	k_{AB}	A	S	N_A	N_B	k_{AB}
Message 1			$11AB$					1?#		
Message 2				1?#					$11AB$	
Message 3							1			$11AB$
Message 4	1	1	$11AB$		$11AB$					
End of run	1	1			$11AB$		1			$11AB$

From the absence of the A's liveness, there exists an attack as stated in Example 3.15: the adversary I has recorded the message $\{M, A, B, \{N_A, M, A, B\}_{K_{AS}}\}$ in an old protocol run, then I can launch a new protocol run by impersonating A.

4.3.10 Security analysis of other protocols

Example 4.23 The KryptoKnight protocol[34] as shown in Fig. 4.25 is a mutual authentication protocol based on symmetric-key cryptography. N_A serves as nonce for entity authentication of B and N_B serves as nonce for entity authentication of A.

$$\begin{aligned}
&\text{Message 1} \quad A \rightarrow B : A, B, N_A \\
&\text{Message 2} \quad B \rightarrow A : B, A, N_B, H(N_A, N_B, B, K_{AB}) \\
&\text{Message 3} \quad A \rightarrow B : A, B, H(N_A, N_B, K_{AB})
\end{aligned}$$

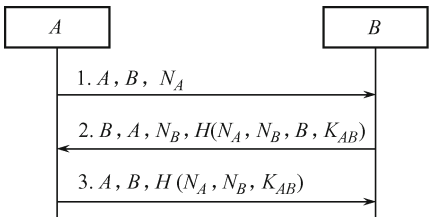

Fig. 4.25 The KryptoKnight mutual authentication protocol.

Notation

A and B are two protocol principals. N_A and N_B are nonces. K_{AB} is a shared long-term key. $H(x_1, x_2, ..., k)$ is a keyed cryptographic hash function.

Premise

K_{AB} is the shared long-term key between A and B, which is initially established by non-cryptographic, and out-of-band techniques; N_A, N_B are nonces generated by A and B respectively.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identities of A, B and a randomly chosen nonce N_A.

2) B randomly chooses a nonce N_B, and sends Message 2 to A to indicate the liveness of B. Message 2 includes the identities of A, B and the hash value $H(N_A, N_B, B, K_{AB})$.

3) Upon receiving Message 2, A gets N_B, and then recalculates and compares $H(N_A, N_B, B, K_{AB})$ with the received hash value. If it matches, then A confirms the liveness of B.

4) In Message 3, A generates and sends the hash value $H(N_A, N_B, K_{AB})$ to B to indicate the liveness of A.

5) Upon receiving Message 3, B recalculates and compares $H(N_A, N_B, K_{AB})$ with the received hash value. If it matches, then B knows that it must be A who has generated and sent the hash value $H(N_A, N_B, K_{AB})$ using their shared long-term key K_{AB}.

Successful execution should convince A and B that both entities are present.

Protocol security analysis

1) In Message 1, from Lemma 4.3, A has the freshness assurance of the TVP N_A.

2) Upon receiving Message 1, B couldn't get any assurance about this protocol.

3) In Message 2, from Lemma 4.3, B has the freshness assurance of the TVP N_B.

4) Upon receiving Message 2, A gets N_B, recalculates $H(N_A, N_B, B, K_{AB})$, and compares the recalculation value with the received hash value. If it matches, from Lemma 4.2, then A knows that it must be B who has generated and sent the hash value $H(N_A, N_B, B, K_{AB})$ including the trusted freshness N_A using their shared long-term key K_{AB}. Hence, the liveness of B is authenticated.

5) Similarly, upon receiving Message 3, B recalculates $H(N_A, N_B, K_{AB})$, and compares the recalculation value with the received hash value. If it matches, from Lemma 4.2, then B knows that it must be A who has generated and sent the hash value $H(N_A, N_B, K_{AB})$ including the trusted freshness N_B

using their shared long-term key K_{AB}. Hence, the liveness of A is authenticated.

Upon termination of the protocol run, A believes that B is alive from comparing the hash value $H(N_A, N_B, K_{AB})$ including the trusted fresh nonce N_A, and B believes that A is alive from comparing the hash value $H(N_A, N_B, K_{AB})$ including the trusted fresh nonce N_B.

The security analysis result in Table 4.24 based on the trusted freshness shows that the KryptoKnight protocol has achieved the mutual authentication objectives as it intends to.

Table 4.24 Security analysis of the KryptoKnight protocol

	A			B		
	B	N_A	N_B	A	N_A	N_B
Message 1		01#			0?#	
Message 2	1	01AB	01AB			01AB
Message 3				1	01AB	
End of run	1			1		

Example 4.24 The Yahalom protocol[35] is a classical authenticated key establishment protocol as shown in Fig. 4.26. The protocol intends to establish a new session key k_{AB} between A and B with the help of a server S, and to achieve mutual authentication.

$$\begin{aligned}
&\text{Message 1} \quad A \rightarrow B: \quad A, N_A \\
&\text{Message 2} \quad B \rightarrow S: \quad B, \{A, N_A, N_B\}_{K_{BS}} \\
&\text{Message 3} \quad S \rightarrow A: \quad \{B, k_{AB}, N_A, N_B\}_{K_{AS}}, \{A, k_{AB}\}_{K_{BS}} \\
&\text{Message 4} \quad I(A) \rightarrow B: \{A, k_{AB}\}_{K_{BS}}, \{N_B\}_{k_{AB}}
\end{aligned}$$

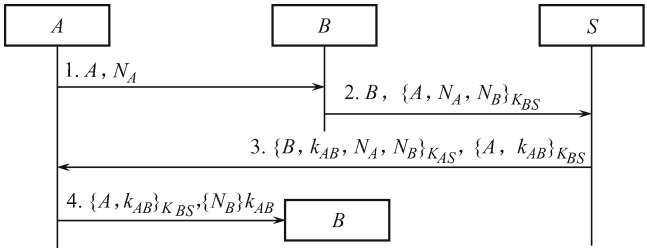

Fig. 4.26 The Yahalom protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. N_A and N_B are nonces. K_{AS} and K_{BS} are shared long-term keys. k_{AB} is a new session key between A and B to be established in this authentication protocol.

Premise

K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively, which are initially established by non-cryptographic, and out-of-band techniques. N_A, N_B are nonces generated by A and B respectively.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identity of A and a randomly chosen nonce N_A.

2) In Message 2, B randomly chooses a nonce N_B, and sends the encryption $\{A, N_A, N_B\}_{K_{BS}}$ to S to show the ownership of the long-term key K_{BS}.

3) Upon receiving Message 2, S gets N_A and N_B.

4) In Message 3, S calculates $\{B, k_{AB}, N_A, N_B\}_{K_{AS}}$, $\{A, k_{AB}\}_{K_{BS}}$ using the shared long-term key K_{AS} and K_{BS} and sends Message 3 to A.

5) Upon receiving Message 3, A gets N_B and the new session key k_{AB} from the encryption $\{B, k_{AB}, N_A, N_B\}_{K_{AS}}$.

6) In Message 4, A encrypts N_B using the new session key k_{AB} to show the liveness of A and the knowledge of k_{AB}.

7) Upon receiving Message 4, B gets the new session key k_{AB} from the encryption $\{A, k_{AB}\}_{K_{BS}}$, and then B checks A's liveness and A's knowledge of k_{AB} via $\{N_B\}_{k_{AB}}$.

Successful execution should convince A and B that both entities are present and k_{AB} is the new session key for A and B.

Protocol security analysis

1) In Message 1, from Lemma 4.3, A has the freshness assurance of the TVP N_A.

2) Upon receiving Message 1, B couldn't get any assurance about this protocol.

3) In Message 2, from Lemma 4.2, both A and B have the confidentiality assurance of N_B. From Lemma 4.3, B has the freshness assurance of the TVP N_B. From Lemma 4.4, B has the association assurance of N_B with A and B.

4) Upon receiving Message 2, S gets N_A and N_B using the long-term key K_{BS}. From Lemma 4.2, S has the confidentiality assurance of N_B, but S couldn't get any assurance about the liveness of B and the freshness of N_A and N_B for there is not any trusted freshness of S in the transformation $\{A, N_A, N_B\}_{K_{BS}}$.

5) Upon receiving Message 3, from Lemma 4.1, A has the liveness assurances of both B and S. From Lemma 4.2, A has the confidentiality assurance of N_B and k_{AB}. From Lemma 4.3, A has the freshness assurance of N_B and k_{AB}. From Lemma 4.4, A has the association assurance of N_A, N_B and k_{AB} with A and B.

6) Upon receiving Message 4, from Lemma 4.1, B has the liveness assurance of A. From Lemma 4.2, B has the confidentiality assurance of k_{AB}.

From Lemma 4.3, B has the freshness assurance of k_{AB} from the trusted freshness N_B. From Lemma 4.4, B has the association assurance of k_{AB} with A and B from the trusted freshness N_B's association with A and B.

Upon termination of this protocol run, the security analysis result in Table 4.25 based on the trusted freshness shows that the Yahalom protocol has achieved the mutual authentication and key establishment objectives.

Table 4.25 Security analysis of the Yahalom protocol

	A					B					S		
	B	S	N_A	N_B	k_{AB}	A	S	N_A	N_B	k_{AB}	N_A	N_B	k_{AB}
Message 1			$01\#$					$0?\#$			$0?\#$		
Message 2				$1?\#$					$11AB$			$1?\#$	
Message 3	1	1	$01AB$	$11AB$	$11AB$								$1?\#$
Message 4						1				$11AB$			
End of run	1				$11AB$	1				$11AB$			

Example 4.25 The TMN protocol[36] is a key establishment protocol based on asymmetric-key cryptography with trusted third party, as illustrated in Fig. 4.27.

$$
\begin{aligned}
&\text{Message 1} \quad A \to S : A, S, B, \{k_A\}_{K_S} \\
&\text{Message 2} \quad S \to B : S, B, A \\
&\text{Message 3} \quad B \to S : B, S, A, \{k_B\}_{K_S} \\
&\text{Message 4} \quad S \to A : S, A, B, V(k_A, k_B)
\end{aligned}
$$

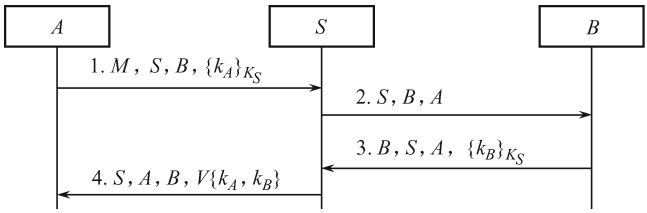

Fig. 4.27 The TMN protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. K_S is a public-key of S. k_A and k_B are randomly chosen input by A and B respectively, and k_B is also the new session key to be established between A and B in the TMN protocol.

The Vernam encryption $V(k_1, k_2)$ is the bit XOR of the two keys k_1 and k_2 where $V(k_1, V(k_1, k_2)) = k_2$. Suppose the randomly input keys k_1 and k_2 are redundancy, hence the receiver could determine whether the received encryption $V(k_1, k_2)$ is correctly decrypted or not.

Premise

K_S and K_S^{-1} are the public-key and the private key of the trusted third party S, which is initially established by non-cryptographic, and out-of-band techniques. k_B is the new session key to be established between A and B in this protocol, and k_B can be recovered by A from the Vernam encryption $V(k_A, k_B)$.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identities of A, S, B and the encryption of a randomly chosen key input k_A under the public-key K_S.

2) Upon receiving Message 1, S gets k_A from the encryption of $\{k_A\}_{K_S}$ using S's private key K_S^{-1}.

3) In Message 2, S notices B launching a new session between A and B by sending the identities of A, B and S.

4) In Message 3, B randomly chooses a new key k_B for this session between A and B, and sends S the encryption of k_B under S's public-key K_S.

5) Upon receiving Message 3, S gets k_B from the encryption of $\{k_B\}_{K_S}$ using S's private key K_S^{-1}.

6) In Message 4, S sends A the new session key k_B via the Vernam encryption $V(k_A, k_B)$.

7) Upon receiving Message 4, A resumes the new session key k_B via the ownership of k_A and the recalculation of $V(k_A, V(k_A, k_B)) = k_B$.

Successful execution should establish a new session key k_B between A and B.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the TVP k_A.

2) Upon receiving Message 1, from Lemma 4.2, B has the confidentiality assurance of the TVP k_A.

3) In Message 2, neither A nor B could get any new assurance about this protocol.

4) In Message 3, from Lemma 4.2 and Lemma 4.3, B has the confidentiality and freshness assurances of the TVP k_B.

5) Upon receiving Message 3, from Lemma 4.2, S has the confidentiality assurance of the TVP k_B. A also has the confidentiality assurance of the TVP k_B.

6) Upon receiving Message 4, from Lemma 4.2, A has the confidentiality assurance of the TVP k_B. From Lemma 4.3, A has the freshness assurance of the TVP k_B from the trusted freshness k_A.

Upon termination of the protocol run, the security analysis result in Table 4.26 shows that: both A and B are not sure whether the opponent principal is present or not, and whether k_B is a new session key for A and B or not.

From the absence of the security properties of an authentication protocol, various attacks could be directly constructed[37].

Table 4.26 Security analysis of the TMN protocol

	A				B			
	B	S	k_A	k_B	A	S	k_A	k_B
Message 1			11#				1?#	
Message 2								
Message 3				1?#				11#
Message 4				11#				
End of run				11#				11#

Example 4.26 (Attack 1 on the TMN protocol) From the absence of A's liveness, the attacker may launch an attack without the presence of A, as illustrated in Fig. 4.28.

$$\begin{aligned}
&\text{Message 1} \quad I(A) \rightarrow S : A, S, B, \{k'_A\}_{K_S} \\
&\text{Message 2} \quad S \rightarrow B : S, B, A \\
&\text{Message 3} \quad B \rightarrow S : B, S, A, \{k_B\}_{K_S} \\
&\text{Message 4} \quad S \rightarrow I(A) : S, A, B, V(k'_A, k_B)
\end{aligned}$$

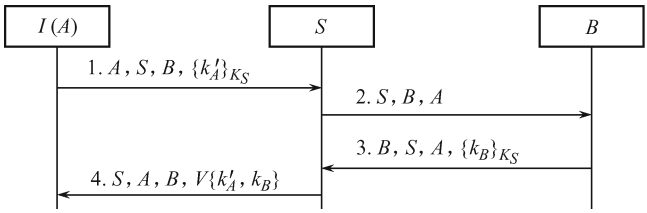

Fig. 4.28 An attack on the TMN protocol by impersonating A.

Notation

$I(A)$ is the adversary I impersonating A.

Premise

k'_A is randomly chosen by I as the nonce to launch a new session between A and B by impersonating A. k_B is randomly chosen by B as the new session key to be established between A and B.

Protocol actions

1) In Message 1, the adversary I randomly chooses a TVP k'_A to launch a new session between A and B by impersonating A.

2) Upon receiving Message 1, S gets k'_A from the encryption of $\{k'_A\}_{K_S}$ using S's private key K_S^{-1}.

3) In Message 2, S notices B launching a new session between A and B by sending the identities of A, B and S.

4) In Message 3, B randomly chooses a new key k_B for this session between A and B (actually between I and B), and sends S the encryption of k_B under S's public-key K_S.

5) Upon receiving Message 3, S gets k_B from the encryption of $\{k_B\}_{K_S}$ using S's private key K_S^{-1}.

6) In Message 4, S sends A the new session key k_B via the Vernam encryption $V(k'_A, k_B)$.

7) Upon receiving Message 4, $I(A)$ resumes the new key k_B via the ownership of k'_A and the recalculation of $V(k'_A, V(k'_A, k_B)) = k_B$.

Upon termination of the attack on the TMN protocol, the adversary I causes B to have false beliefs: B has completed a successful protocol run with A, and is sharing a new session key k_B with A, but actually shares the key k_B with I. Furthermore, B concludes that subsequently messages could be encrypted using k_B and safely transmitted to A (actually known by I), whereas in fact, A knows nothing about the key establishment.

Example 4.27 (Attack 2 on the TMN protocol) From the absence of the B's liveness, the attacker may launch an attack without the presence of B, as illustrated in Fig. 4.29.

$$
\begin{aligned}
&\text{Message 1} \quad A \to S: \quad A, S, B, \{k_A\}_{K_S} \\
&\text{Message 2} \quad S \to I(B): S, B, A \\
&\text{Message 3} \quad I(B) \to S: B, S, A, \{k'_B\}_{K_S} \\
&\text{Message 4} \quad S \to A: \quad S, A, B, V(k_A, k'_B)
\end{aligned}
$$

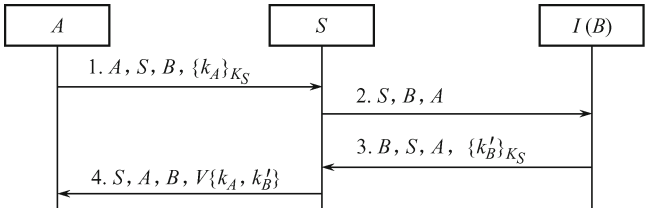

Fig. 4.29 An attack on the TMN protocol by impersonating B.

Notation

$I(B)$ is the adversary I impersonating B.

Premise

k'_B is randomly chosen by I as the new session key to be established between A and B by impersonating B.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identities of A, S, B and the encryption of a randomly chosen key input k_A under S's public-key K_S.

2) Upon receiving Message 1, S gets k_A from the encryption of $\{k_A\}_{K_S}$ using S's private key K_S^{-1}.

3) In Message 2, S notices B launching a new session between A and B by sending the identities of A, B and S.

4) In Message 3, the adversary I randomly chooses a TVP k'_B as the new session key between A and B by impersonating B (actually between I and B), and sends S the encryption of k'_B under S's public-key K_S.

5) Upon receiving Message 3, S gets k'_B from the encryption of $\{k'_B\}_{K_S}$ using S's private key K_S^{-1}.

6) In Message 4, S sends A the new session key k'_B via the Vernam encryption $V(k_A, k'_B)$.

7) Upon receiving Message 4, A resumes the new key k'_B via the ownership of k_A and the recalculation of $V(k_A, V(k_A, k'_B)) = k'_B$.

Upon termination of the attack on the TMN protocol, the adversary I causes A to have false beliefs: A has completed a successful protocol run with B, and is sharing a new session key k'_B with B, but actually shares the key k'_B with I. Furthermore, A concludes that subsequently messages could be encrypted using k'_B and safely transmitted to B (actually known by I), whereas in fact, B knows nothing about the key establishment.

Example 4.28 (Attack 3 on the TMN protocol) From the absence of the key association assurance of k_B, the adversary I could get the secret new session key k_B which is intended only for A and B, as illustrated in Fig. 4.30.

Message 1	$A \to I(A):$	$A, S, B, \{k_A\}_{K_S}$
Message 1′	$I(A) \to S:$	$A, S, B, \{k_A\}_{K_S}$
Message 2′	$S \to I(B):$	S, B, A
Message 3′	$I(B) \to S:$	$B, S, A, \{k'_B\}_{K_S}$
Message 4′	$S \to I(A):$	$S, A, B, V(k_A, k'_B)$
		(Now I knows k_A from $V(k'_B, V(k_A, k'_B)) = k_A$)
Message 1″	$A \to S:$	$A, S, B, \{k_A\}_{K_S}$
Message 2	$S \to B:$	S, B, A
Message 3	$B \to S:$	$B, S, A, \{k_B\}_{K_S}$
Message 4″	$S \to I(A):$	$S, A, B, V(k_A, k_B)$
		(Now I knows k_B from $V(k_A, V(k_A, k_B)) = k_B$)
Message 4	$I(S) \to A:$	$S, A, B, V(k_A, k_B)$

Notation

$I(A)$ or $I(B)$ is the adversary I impersonating A and B independently.

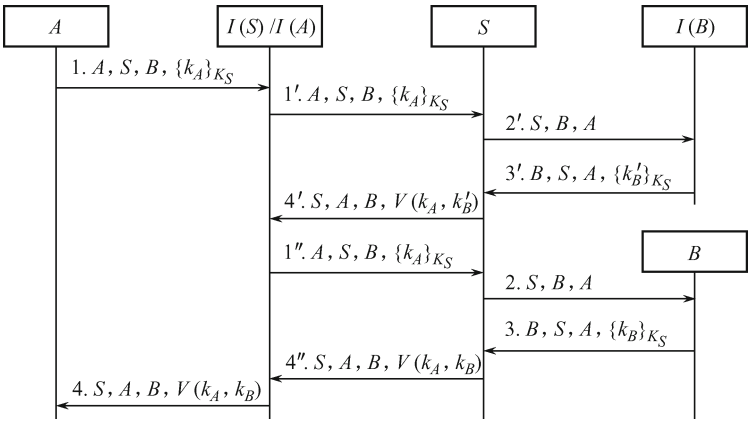

Fig. 4.30 An attack on the TMN protocol to get the secret new session key k_B between A and B by impersonating both A and B.

Premise

k'_B is randomly chosen by I as the new session key to be established between A and B by impersonating B.

Protocol actions

1) In Message 1, A launches a new protocol run by sending the identities of A, S, B and the encryption of a randomly chosen key input k_A under the public-key K_S of S.

2) The adversary I intercepts Message 1 and replays this message as Message $1'$ to S by impersonating A.

3) In Message $2'$, S notices B launching a new session between A and B by sending the identities of A, B and S.

4) The adversary I intercepts Message $2'$, and randomly chooses a TVP k'_B as the new session key between A and B by impersonating B (actually between I and A), and sends S the encryption of k'_B under S's public-key K_S.

5) Upon receiving Message $3'$, S gets k'_B from the encryption of $\{k'_B\}_{K_S}$ using S's private key K_S^{-1}.

6) In Message $4'$, S sends A (Indeed I) the new session key k'_B via the Vernam encryption $V(k_A, k'_B)$.

7) Upon receiving Message $4'$, the adversary I intercepts Message $4'$ and resumes k_A via the ownership of the TVP k'_B and the recalculation of $V(k'_B, V(k_A, k'_B)) = k_A$.

8) In Message $1''$, the adversary I forwards Message 1 to S, indicating a new protocol run between A and B.

9) Upon receiving Message $1''$, S gets k_A from the encryption of $\{k_A\}_{K_S}$ using S's private key K_S^{-1}.

10) In Message 2, S notices B the new session between A and B by sending the identities of A, B and S.

11) In Message 3, B randomly chooses a new key k_B for this session between A and B, and sends S the encryption of k_B under S's public-key K_S.

12) Upon receiving Message 3, S gets k_B from the encryption of $\{k_B\}_{K_S}$ using S's private key K_S^{-1}.

13) In Message $4''$, S sends A the new session key k_B via the Vernam encryption $V(k_A, k_B)$.

14) The adversary I intercepts Message $4''$, and resumes the new key k_B via the ownership of k_A and the recalculation of $V(k_A, V(k_A, k_B)) = k_B$. Up to now, I knows the session key k_B which is intended to be secret and only known by A and B.

15) In Message 4, the adversary I forwards Message $4''$ as Message 4 to A.

16) Upon receiving Message 4, A resumes the new key k_B via the ownership of k_A and the recalculation of $V(k_A, V(k_A, k_B)) = k_B$.

Upon termination of the attack on the TMN protocol, the adversary I has gotten the secret new session key k_B between A and B, which is intended to be secret and only known by A and B. As a result of this, the adversary I could get the subsequent sensitive encryption communicated between A and B, via using the decryption key k_B.

Note that Message $1''$ and Message 1 are actually the same message from A to S, and the adversary I has intercepted this message. Similar case exists in Message $4''$ and Message 4.

Example 4.29 The big mouth frog protocol[38] is a key transport protocol based on symmetric-key cryptography with trusted third party, as illustrated in Fig. 4.31.

$$\begin{array}{ll} \text{Message 1} & A \to S : A, \{B, k_{AB}\}_{K_{AS}} \\ \text{Message 2} & S \to B : \{A, k_{AB}\}_{K_{BS}} \end{array}$$

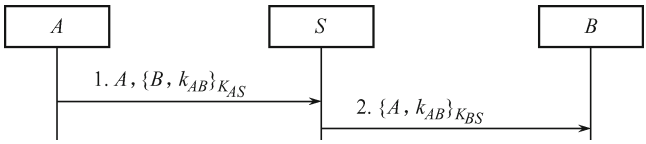

Fig. 4.31 The big mouth frog protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. k_{AB} is randomly chosen by A as the new session key to be established between A and B.

Premise

K_{AS} and K_{BS} are shared long-term keys between A and S, and B and S respectively, which are initially established by non-cryptographic, and out-

of-band techniques.

Protocol actions

1) In Message 1, A randomly chooses a new key k_{AB} for this session between A and B, and sends S the encryption of $\{B, k_{AB}\}_{K_{AS}}$ using the shared long-term key K_{AS} to indicate that A wants to establish a subsequent communication key with B, and k_{AB} is intended for A and B.

2) Upon receiving Message 1, S gets k_{AB} from the encryption of $\{B, k_{AB}\}_{K_{AS}}$ using the shared long-term key K_{AS}.

3) In Message 2, S notices B launching a new session between A and B by sending the encryption $\{A, k_{AB}\}_{K_{BS}}$.

4) Upon receiving Message 2, B gets k_{AB} from the encryption of $\{A, k_{AB}\}_{K_{BS}}$ using the shared long-term key K_{BS}.

Successful execution should establish a new session key k_{AB} between A and B.

Protocol security analysis

1) In Message 1, from Lemma 4.2 and Lemma 4.3, A has the confidentiality and freshness assurances of the TVP k_{AB}. From Lemma 4.4, A has the association assurance of the k_{AB} with B from the explicitly mentioned principal name and the trusted freshness k_{AB}. From Lemma 4.4, A also has the association assurance of the k_{AB} with A since only A could have the new session key k_{AB} which is encrypted under the shared long-term key K_{AS}.

2) Upon receiving Message 1, from Lemma 4.2, B has the confidentiality assurance of the TVP k_{AB}.

3) In Message 2, B couldn't get any new assurance about this protocol since there is not a trusted freshness in the encryption $\{A, k_{AB}\}_{K_{BS}}$.

Upon termination of the protocol run, the security analysis result in Table 4.27 shows that: both A and B are not sure whether the opponent principal is present or not, and B is not sure whether k_{AB} is a fresh session key for A and B or not. From the absence of the security properties of an authentication protocol, various attacks could be directly constructed.

Table 4.27 Security analysis of the big mouth frog protocol

	A			B		
	B	S	k_{AB}	A	S	k_{AB}
Message 1			11AB			1?#
Message 2						
End of run			11AB			1?#

Example 4.30 (Attack on the big mouth frog protocol) From the absence of the A's liveness, the attacker may launch an attack without the presence of A, as shown in Fig. 4.32. Suppose the adversary I has recorded the message $\{A, \{B, k'_{AB}\}_{K_{AS}}\}$ in an old protocol run, then I can launch a new protocol

run by impersonating A.

$$\begin{array}{ll} \text{Message 1} & I(A) \rightarrow S : A, \{B, k'_{AB}\}_{K_{AS}} \\ \text{Message 2} & S \rightarrow B : \quad \{A, k'_{AB}\}_{K_{BS}} \end{array}$$

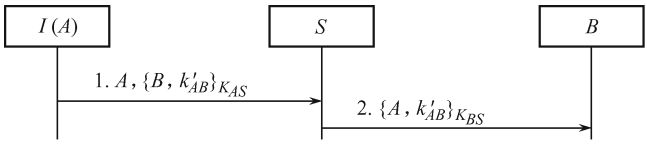

Fig. 4.32 An attack on the big mouth frog protocol.

Notation

$I(A)$ is an adversary I impersonating A.

Premise

k'_{AB} is a compromised session key between A and B. The adversary I has recorded $\{A, \{B, k_{AB}\}_{K_{AS}}\}$ in Message 1 of an old protocol run.

Protocol actions

1) In Message 1, the adversary I replays the recorded message $\{A, \{B, k_{AB}\}_{K_{AS}}\}$ to S to indicate that A wants to establish a subsequent communication key k'_{AB} with B, and k'_{AB} is intended for A and B.

2) Upon receiving Message 1, S gets k'_{AB} from the encryption of $\{B, k'_{AB}\}_{K_{AS}}$ using the shared long-term key K_{AS}.

3) In Message 2, S notices B launching a new session between A and B by sending the encryption $\{A, k'_{AB}\}_{K_{BS}}$.

Upon termination of the attack on the big mouth frog protocol, the adversary I causes B to have false beliefs: B has completed a successful protocol run with A, and is sharing a new session key k'_{AB} with A, but actually the key k'_{AB} is known by I. Furthermore, B concludes that subsequently messages could be encrypted using k'_{AB} and safely transmitted to A (actually known by I), whereas in fact, A knows nothing about the key establishment.

From the absence of the B's liveness, the attacker may also intercept the protocol run, and cause A to have false beliefs: A has completed a successful protocol run with A, and is sharing a new session key k'_{AB} with B, but actually B knows nothing about the key establishment.

Example 4.31 A electronic voting protocol is a voting protocol with deniable authentication for general elections[39, 40]. Fig. 4.33 shows that the voting center S collects each voter A_i's vote V_{A_i} and believes that the vote V_{A_i} is not repudiated via the use of the fresh nonce N_S; Every voter A_i is notified of the voting result RE and the voter A_i believes that the vote result RE is

noot repudiated via the use of the fresh nonce N_{A_i}.

Message 1 $S \rightarrow A_i : N_S$
Message 2 $A_i \rightarrow S : A_i, N_{A_i}, V_{A_i}, H(N_S, S_{A_i}, V_{A_i})$
Message 3 $S \rightarrow A_i : RE, H(N_{A_i}, S_{A_i}, RE)$

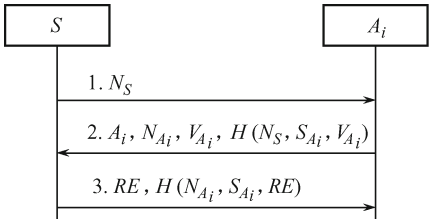

Fig. 4.33 An electronic voting protocol.

Notation

A_i is a voter who casts a vote in this electronic voting protocol, and S is a trusted third party who works as a voting center. S_{A_i} is the shared long-term key between the voter A_i and the voting center S. V_{A_i} is the vote which is sent by every voter A_i at the beginning of voting, and RE is the voting result collected by the voting center S.

Premise

The shared long-term S_{A_i} is initially established by non-cryptographic, and out-of-band techniques. H (nonce, key, data) is a keyed hash function with three input values. N_S is a nonce randomly chosen by S. N_{A_i} is a nonce randomly chosen by A_i.

Protocol actions

1) In Message 1, S launches a new electronic voting process by sending each voter A_i a randomly chosen fresh nonce N_S.

2) Upon receiving Message 1, every voter A_i gets N_S.

3) In Message 2, the voter A_i randomly chooses a nonce N_{A_i} for this voting, sends the identity of A_i, the nonce N_{A_i}, and A_i's vote V_{A_i} to S. A_i wants to show the freshness and the ownership of the vote V_{A_i} to S via a keyed hash function under the shared long-term S_{A_i}.

4) Upon receiving Message 2, S gets the vote V_{A_i} of the voter A_i and checks the validity of V_{A_i} via the ownership of S_{A_i} and the recalculation of $H(N_S, S_{A_i}, V_{A_i})$. Then, S summarizes all the votes of every voter and works out the voting result RE. Every voter A_i could not deny its sending vote V_{A_i} to S for the hash value $H(N_S, S_{A_i}, V_{A_i})$ including the trusted freshness N_S.

5) In Message 3, S announces the voting result RE to every voter A_i and wants to show the freshness and the ownership of the voting result RE to every A_i via a keyed hash function under the shared long-term key S_{A_i}.

6) Upon receiving Message 3, the voter A_i checks the validity of RE via the ownership of S_{A_i} and the recalculation of $H(N_{A_i}, S_{A_i}, RE)$. S could not deny its sending the voting result RE, which is summarized by S, to every voter A_i for the hash value $H(N_{A_i}, S_{A_i}, RE)$ includes the trusted fresh nonce N_{A_i} generated by A_i.

Successful execution should work out a new voting result RE, and neither the voter A_i nor the voting center S could deny the participation of this electronic voting.

Protocol security analysis

1) In Message 1, from Lemma 4.3, S has the freshness assurance of the TVP N_S, that is, N_S is the trusted freshness of S.

2) Upon receiving Message 1, B couldn't get any assurance about this protocol.

3) In Message 2, from Lemma 4.3, A_i has the freshness assurance of the TVP N_{A_i} and the vote V_{A_i}. From Lemma 4.2, A_i knows that the vote V_{A_i} is open.

4) Upon receiving Message 2, from Lemma 4.3, S has the freshness assurance of the vote V_{A_i}. From Lemma 4.1, S believes that it must be A_i who has generated the message $H(N_S, S_{A_i}, V_{A_i})$ using the shared long-term S_{A_i}. From Lemma 4.2, S knows that the vote V_{A_i} is open.

5) In Message 3, from Lemma 4.3, S has the freshness assurance of the voting result RE. From Lemma 4.2, S knows that the vote RE is open. From Lemma 4.4, S has the association assurance of the voting result RE with S.

6) Upon receiving Message 3, from Lemma 4.2, A_i knows that the voting result RE is open. From Lemma 4.3, A_i has the freshness assurance of the voting result RE. From Lemma 4.1 and Lemma 4.4, A_i has the liveness assurance of S and the association assurance of the vote V_{A_i} with S, since only S could generate $H(N_{A_i}, S_{A_i}, RE)$ using the shared long-term S_{A_i}, hence $H(N_{A_i}, S_{A_i}, RE)$ is associated with the protocol run with S, and S is present.

The security analysis result in Table 4.28 based on the trusted freshness shows that the electronic voting protocol has achieved the general elections with deniable authentication objectives as it intends to. The voter A_i could not deny A_i's vote V_{A_i} and the voting center S could not deny S's announcement of the voting result RE. However, the voter A_i can only believe that RE is a recently announced voting result by S, but A_i does not know whether the voting result RE is associated with each V_{A_i} or not.

Table 4.28 Security analysis of the electronic voting protocol

	A_i					S				
	S	N_S	N_{A_i}	V_{A_i}	RE	A_i	N_S	N_{A_i}	V_{A_i}	RE
Message 1							01#			
Message 2			01#	0?#		1		01#	01#	
Message 3	1				01S					01#
End of run	1				01S	1				01#

Example 4.32 A fair non-repudiation protocol is based on a trusted third party[39, 40]. In the protocol as shown in Fig. 4.34, the principal A sends message m to the principal B with the help of a trusted third party S. At the end of the protocol, A could deny A's sending of the message m, B could not deny B's receiving the message m.

$$\begin{aligned}
&\text{Message 1} \quad A \to B : f_1, B, L, \{m\}_{k'}, \{f_1, B, L, \{m\}_{k'}\}_{K_A^{-1}} \\
&\text{Message 2} \quad B \to A : f_2, A, L, \{m\}_{k'}, \{f_2, A, L, \{m\}_{k'}\}_{K_B^{-1}} \\
&\text{Message 3} \quad A \to S : f_3, B, L, k', \{f_3, B, L, k'\}_{K_A^{-1}} \\
&\text{Message 4} \quad S \to A : f_4, A, B, L, k', \{f_4, A, B, L, k'\}_{K_S^{-1}} \\
&\text{Message 5} \quad S \to B : f_4, A, B, L, k', \{f_4, A, B, L, k'\}_{K_S^{-1}}
\end{aligned}$$

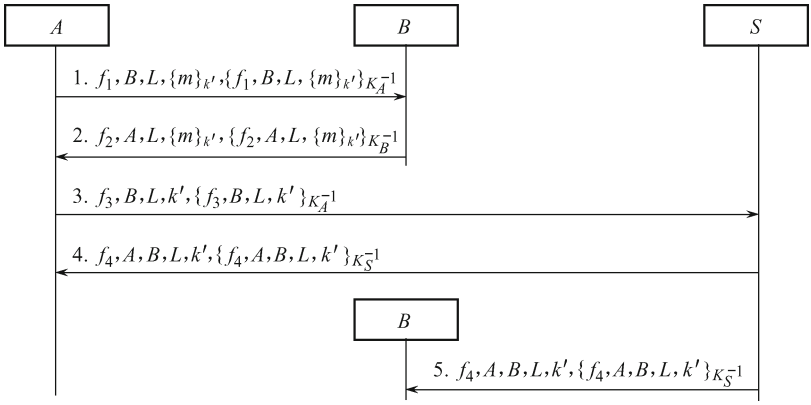

Fig. 4.34 A fair non-repudiation protocol.

Notation

A and B are two protocol principals, and S is a trusted third party. f_i is a message tag to indicate the message step, i.e., data type. L is the life of this protocol run. m is a sensitive data to be sent to B by A. k' is a randomly chosen temporary key for this protocol run by A. K_A and K_A^{-1}, K_B and K_B^{-1}, K_S and K_S^{-1} are the public, and private key pairs of the principals A, B and S respectively.

Premise

The public and private long-term key K_A and K_A^{-1}, K_B and K_B^{-1}, K_S and K_S^{-1} are initially established by non-cryptographic, and out-of-band techniques.

Protocol actions

1) In Message 1, A randomly chooses a temporary key k', and sends the message tag f_1, the opponent partner's identity B, and the life circle L

with the encryption $\{m\}_{k'}$ of the data m to B to launch a new protocol run between A and B. $\{f_1, B, L, \{m\}_{k'}\}_{K_A^{-1}}$ is signed by A's private key K_A^{-1} to indicate that Message 1 is from A.

2) Upon receiving Message 1, B verifies the signature $\{f_1, B, L, \{m\}_{k'}\}_{K_A^{-1}}$ using A's public-key K_A and gets the encryption $\{m\}_{k'}$ of the data m. But B can not confirm whether the received message is new generated by A or not, hence also B can't confirm whether A is present or not.

3) In Message 2, B sends the message tag f_2, the opponent partner's identity A, and the life circle L with the encryption $\{m\}_{k'}$ of the data m to A. $\{f_2, A, L, \{m\}_{k'}\}$ is signed by B's private key K_B^{-1} to indicate that Message 2 is from B.

4) Upon receiving Message 2, A checks the validity of $\{m\}_{k'}$ using B's public-key K_B. If it is right, then it must be B who has gotten $\{m\}_{k'}$ using A's public-key K_A and signed $\{f_2, A, L, \{m\}_{k'}\}$ using B's private key K_B^{-1}.

5) In Message 3, A sends the message tag f_3, the opponent partner's identity B, the life circle L and the temporary key k' to S. $\{f_3, B, L, k'\}$ is signed by A's private key K_A^{-1} to indicate that Message 3 is from A.

6) Upon receiving Message 3, S gets and checks k' from the signature $\{f_3, B, L, k'\}_{K_A^{-1}}$ using A's public-key K_A. Hence, A could not deny A's sending of k'.

7) In Message 4 and Message 5, S sends the message tag f_4, the protocol partners' identities A and B, the life circle L and the randomly chosen temporary key k' to A and B respectively. $\{f_4, A, B, L, k'\}$ is signed by the trusted third party S's private key K_S^{-1} to indicate that Message 4 is from S and the temporary key k' has been checked by S.

8) Upon receiving Message 4, A gets k' from the signature $\{f_4, A, B, L, k'\}_{K_S^{-1}}$ using S's public-key K_S and checks the validity of k'.

9) Upon receiving Message 5, B gets and checks k' from the signature $\{f_4, A, B, L, k'\}_{K_S^{-1}}$ using S's public-key K_S. Hence, B can get the data m from the encryption $\{m\}_{k'}$ via using k'.

Upon termination of the protocol run, from (b) and (f), A could not deny her sending of the message m, and from (d) and (i), B can get the message m. However, B could deny B's receiving of the message m, since no witness shows that B has gotten k', hence the message m.

Protocol security analysis

1) In Message 1, from Lemma 4.2, A has the confidentiality assurance of the temporary key k' and the data m. From Lemma 4.3, A has the freshness assurance of the temporary key k'. That is, k' is the trusted freshness of A.

2) Upon receiving Message 1, from Lemma 4.2 and Lemma 4.3, B has the confidentiality assurance of the temporary key k' and the data m.

3) Upon receiving Message 2, from Lemma 4.1, A has the liveness assurance of B. From Lemma 4.4, A has the association assurance of k' and m with B since Message 2 could not be a replay one, and only B could generate

$\{f_2, A, L, \{m\}_{k'}\}_{K_B^{-1}}$ using B's private key K_B^{-1}. From Lemma 4.4, A has the association assurance of k' and m with A, since the identity of A has been explicitly indicated in $\{f_2, A, L, \{m\}_{k'}\}_{K_B^{-1}}$, so k' and m are also associated with A.

4) In Message 3, from Lemma 4.2, A knows that k' is open.

5) After Message 3 is sent, from Lemma 4.2, B also knows that k' is open according to the protocol.

6) Upon receiving Message 4, from Lemma 4.1, A has the liveness assurance of S since Message 4 could not be a replay one, and only S could generate $\{f_4, A, B, L, k'\}_{K_S^{-1}}$ using its private key K_S^{-1}, hence S is present.

7) Upon receiving Message 5, B couldn't get any assurance about this protocol for B has not gotten any trusted freshness identifier.

The security analysis result in Table 4.29 based on the trusted freshness shows that from the legitimate participant A's point of view, B is present and the data m is open, fresh and associated with both A and B, and from the legitimate participant B's point of view, A is not present and the data m is open, and B could not achieve the association assurance of m with both A and B.

Table 4.29 Security analysis of the fair non-repudiation protocol

	A				B			
	B	S	k'	m	A	S	k'	m
Message 1			11#	11#			1?#	1?#
Message 2	1		11AB	11AB				
Message 3			01AB	01AB			0?#	0?#
Message 4		1						
Message 5								
End of run	1	1	01AB	01AB			0?#	0?#

Example 4.33 (Attack on the fair non-repudiation protocol) From the absence of A's liveness, the attacker may launch an attack without the presence of A, as shown in Fig. 4.35. Suppose the adversary I has recorded the message $\{f_1, B, L, \{m\}_{k''}, \{f_1, B, L, \{m\}_{k''}\}_{K_A^{-1}}\}$ in an old protocol run, then I can launch a new protocol run by impersonating A.

$$\begin{aligned}
&\text{Message 1} \quad I(A) \to B : f_1, B, L, \{m\}_{k''}, \{f_1, B, L, \{m\}_{k''}\}_{K_A^{-1}} \\
&\text{Message 2} \quad B \to I(A) : f_2, A, L, \{m\}_{k''}, \{f_2, A, L, \{m\}_{k''}\}_{K_B^{-1}} \\
&\text{Message 3} \quad I(A) \to S : f_3, B, L, k'', \{f_3, B, L, k''\}_{K_A^{-1}} \\
&\text{Message 4} \quad S \to I(A) : f_4, A, B, L, k'', \{f_4, A, B, L, k''\}_{K_S^{-1}} \\
&\text{Message 5} \quad S \to B : \quad f_4, A, B, L, k'', \{f_4, A, B, L, k''\}_{K_S^{-1}}
\end{aligned}$$

Notation

$I(A)$ is an adversary I impersonating A.

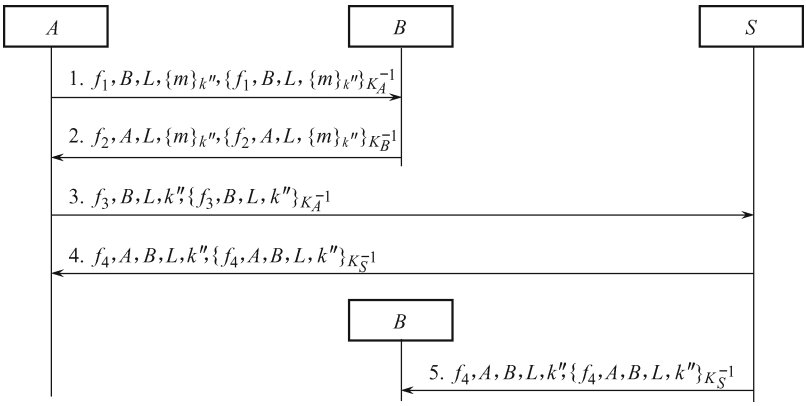

Fig. 4.35 An attack on the fair non-repudiation protocol.

Premise

The adversary I has recorded $\{f_1, B, L, \{m\}_{k''}, \{f_1, B, L, \{m\}_{k''}\}_{K_A^{-1}}\}$ in Message 1 and $\{f_3, B, L, k'', \{f_3, B, L, k''\}_{K_A^{-1}}\}$ in Message 3 of an old protocol run.

Protocol actions

1) In Message 1, the adversary I replays the recorded message $\{f_1, B, L, \{m\}_{k''}, \{f_1, B, L, \{m\}_{k''}\}_{K_A^{-1}}\}$ to B to indicate that A wants to send a sensitive data m to B.

2) Upon receiving Message 1, B verifies the signature $\{f_1, B, L, \{m\}_{k''}\}_{K_A^{-1}}$ and gets the encryption $\{m\}_{k''}$ of the data m.

3) In Message 2, B sends the message $\{f_2, A, L, \{m\}_{k''}\}_{K_B^{-1}}$ to A (actually it is the adversary I).

4) In Message 3, the adversary I replays the recorded message $\{f_3, B, L, k'', \{f_3, B, L, k''\}_{K_A^{-1}}\}$ to S.

5) Upon receiving Message 3, S gets and checks k'' from the signature $\{f_3, B, L, k''\}_{K_A^{-1}}$ using A's public-key K_A.

6) In Message 4 and Message 5, S signs $\{f_4, A, B, L, k''\}$ using S's private key K_S^{-1} and sends it to A and B respectively.

7) Upon receiving Message 4, the adversary I intercepts the message intended for A.

8) Upon receiving Message 5, B gets and checks k'' from the signature $\{f_4, A, B, L, k''\}_{K_S^{-1}}$ using S's public-key K_S.

Upon termination of this attack on the fair non-repudiation protocol, the adversary I causes B to have false beliefs: B has completed a successful protocol run with A, B has received the message m sent from A, and A could not deny the sending of m to B.

In real life, the freshness of non-repudiation record is important. For example, suppose a car agent has ordered 1000 cars from the General Motors

Corporation before, if the adversary can simply replay this order record without freshness guarantee, then the General Motors Corporation may check and accept this order, since the replayed order record may be regarded as non-repudiation assurance.

References

[1] Needham RM, Schroeder MD (1978) Using Encryption for Authentication in Large Network of Computers. Communication of the ACM 21(12): 993 – 999

[2] Feige U, Fiat A, Shamir A (1987) Zero Knowledge Proofs of Identify. In: STOC'87 Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, 25 – 27 May 1987

[3] Miller SP, Neuman BC, Schiller JI, Saltzer JH (1987) Kerberos Authentication and Authorization System. Paper Presented at the Project Athena Technical Plan Section E.2.1. MIT, Boston

[4] CCITT (1987) CCITT Draft Recommendation X.509. The Directory-Authentication Framework (Version 7), New York

[5] Woo TYC, Lam SS (1992) Authentication for Distributed Systems. Computer 25(1): 39 – 52

[6] Kaufman C (1993) Distributed Authentication Security Service, RFC 1507. http://www.ietf.org/rfc/rfc1507.txt. Accessed 7 Sept 2010

[7] Okamoto T (1993) Provably Secure and Practical Identification Schemes and Corresponding Signature Scheme. In: CRYPTO'92 Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara 16 – 20 Aug 1992. Lecture Notes in Computer Science, vol 740, pp 31 – 53, Springer

[8] IBM Zurich Laboratory (1995) Internet Keyed Payments Protocol (IKP). http://www.zurich.ibm.com /Technology/Security/extern/ecommerce/spec. Accessed 30 June 2010

[9] Lowe G (1995) An Attack on the Needham-Schroeder Public-key Authentication Protocol. Information Processing Letters 56(3): 131 – 133

[10] Abadi M, Needham R (1996) Prudent Engineering Practice for Cryptographic Protocols. IEEE Transactions on Software Engineering 22(1): 6 – 15

[11] Freier AO, Karlton P, Kocher PC (1996) The SSL Protocol Version 3.0. http://wp.netscape.com/eng/ssl3/draft302.txt. Accessed 18 Nov 1996

[12] Clark J and Jacob J (1997) A Survey of Authentication Protocol Literature: Version 1.0. http://www.win.tue.nl/∼ecss/downloads/clarkjacob.pdf. Accessed Nov 2010

[13] SET. Secure Electronic Transaction. The SET Standard Specification. http://www.setco.org/set-specifications. Accessed May 1997

[14] Harkins D, Carrel D (1998) The Internet Key Exchange Protocol (IKE), RFC 2409. http://www.ietf.org/rfc/rfc2409.txt. Accessed 12 Dec 2010

[15] ANSI/IEEE Std 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Sept 1999

[16] Burrows M, Abadi M, Needham R (1990) A Logic of Authentication. ACM Transactions on Computer Systems 8(1): 18 – 36

[17] Bellare M, Rogaway P (1993) Entity Authentication and Key Distribution. In: CRYPTO'93 Proceedings of the 13th Annual International Cryptology

Conference on Advances in Cryptology, Santa Barbara, 22 – 26 Aug 1993. Lecture Notes in Computer Science, vol 773, pp 232 – 249, Springer

[18] Lowe G (1999) Towards a Completeness Result for Model Checking of Security Protocols. Journal of Computer Security 7(2 – 3): 89 – 146

[19] Canetti R, Krawczy H (2001) Analysis of Key-exchange Protocols and Their Use for Building Secure Channels. In: EUROCRYPT'01 Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, Innsbruck, 6 – 10 May 2001. Lecture Notes in Computer Science, vol 2045, pp 453 – 474, Springer

[20] Blanchet B (2006) A Computationally Sound Mechanized Prover for Security Protocols. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy, Berkeley/Oakland, 21 – 24 May 2006

[21] Datta A, Derek A, Mitchell JC, Roy A (2007) Protocol Composition Logic (PCL). Electronic Notes in Theoretical Computer Science 172: 311 – 358.

[22] Mao W (2004) Modern Cryptography: Theory and Practice. Prentice Hall, New Jersey

[23] Dolev D, Yao AC (1983) On the Security of Public Key Protocols. IEEE Transactions on Information Theory 29(2): 198 – 208

[24] Bellare M, Rogaway P (1993) Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In: CCS'93 Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, 3 – 5 Nov 1993

[25] Goldwasser S, Micali S (1984) Probabilistic Encryption. Journal of Computer and System Sciences 28(2): 270 – 299

[26] Dong L, Chen K, Zheng Y, Hong X (2008) The Guarantee of Authentication Protocol Security. Journal of Shanghai JiaoTong University 42(4): 518 – 522

[27] Otway D, Rees O (1987) Efficient and Timely Mutual Authentication. Operating Systems Review 21(1): 8 – 10

[28] Diffie W, Hellman ME (1976) New Directions in Cryptography. IEEE Transactions on Information Theory 22(6): 644 – 654.

[29] Menezes A, van Oorschot P, Vanstone S (1996) Handbook of Applied Cryptography. CRC Press, New York

[30] Matsumoto T, Takashima Y, Imai H (1986) On Seeking Smart Public-key Distribution Systems. Trans. IECE Japan 69(2): 99 – 106.

[31] Denning DE, Sacco GM (1981) Timestamps in Key Distribution Protocols. Communication of the ACM 24(8): 533 – 536

[32] Woo TYC, Lam SS (1994) A Lesson on Authentication Protocol Design. ACM Operating Systems Review 28(3): 24 – 37

[33] Neuman BC, Stubblebine SG (1993) A Note on the Use of Timestamps as Nonces. Operating Systems Review 27(2): 10 – 14

[34] Bird R, Gopal I, Herzberg A, Janson P, Kutten S, Molva R, Yung M (1995) The KryptoKnight Family of Light-weight Protocols for Authentication and Key Distribution. IEEE/ACM Transactions on Networking 3(1): 31 – 41

[35] Yahalom. http://www.lsv.ens-cachan.fr/spore/yahalom.pdf. Accessed 12 May 2011

[36] Tatebayashi M, Matsuzaki N, Newman D (1989) Key Distribution Protocol for Digital Mobile Communication Systems. In: CRYPTO'93 Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, 20 – 24 Aug 1989. Lecture Notes in Computer Science, vol 435, pp 324 – 334, Springer

[37] Lowe G, Roscoe B (1997) Using CSP to Detect Errors in the TMN Protocol. IEEE Transactions on Software Engineering 23(10): 659 – 669

[38] Tanenbaum AS (2001) Computer Networks, 3rd edn. Prentice Hall, New Jersey

[39] Zhou J, Gollmann D (1996) A Fair Non-repudiation Protocol. In: Proceedings of 1996 IEEE Symposium on Security and Privacy, Oakland, 6 – 8 May 1996

[40] Zhou J (1996) Non-repudiation. PhD Dissertation, University of London