Cryprography and Network Security, PART II: Key Exchange Protocols

Timo Karvi

10.2012

Building a Key Establishement Protocol

- We now explicate an attempt to design a good protocol. The situation is as follows:
- We assume that there is a set of users, any two of whom may wish to establish a new key for use in securing their subsequent communications through cryptography. Such a key is known as a session key.
- In order to achieve their aim the users interact with an entity called the server which will also engage in the protocol. All users trust the server to execute the protocol faithfully and not to engage in any other activity that will deliberately compromise their security. Furthermore, the server is trusted to generate the new key and to do so in such a way that it is sufficiently random to prevent an attacker gaining any useful information about it.
- Thus there are two users, A and B, and the trusted server S. The aim of the protocol is to establish a new secret key K_{AB} between A and B. The role of S is to generate the key and transport it to A and B.

First Attempt

Our first attempt is the following naïve protocol with three messages:

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: K_{AB}
3. $A \longrightarrow B$: K_{AB}, A

Usually we make the following security assumption 1: *The adversry is able to eavesdrop on all messages sent in a cryptographic protocol.*

If we assume this assumption, we see that the protocol is vulnerable, because the adversary can take the secret key. We must thus assume that A and S as well as B and S share a secret key.

Second Attempt I

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_{AB}\}_{K_{AS}}, \{K_{AB}\}_{K_{BS}}$
3. $A \longrightarrow B$: $\{K_{AB}\}_{K_{BS}}, A$

This protocol is as insecure in an open environment as our first attempt, but for a completely different reason.

Security Assumption 2:
The adversary is able to alter all messages sent in a cryptographic protocol using any information available. In addition the adversary can re-route any message to any other principal. This includes the ability to generate and insert completely new messages.

By applying the second security assumption it is possible to break the protocol without breaking the cipher: The attack proceeds as follows:

Second Attempt II

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_{AB}\}_{K_{AS}}, \{K_{AB}\}_{K_{BS}}$
3. $A \longrightarrow C$: $\{K_{AB}\}_{K_{BS}}, A$
4. $C \longrightarrow B$: $\{K_{AB}\}_{K_{BS}}, D$

The adversary C simply intercepts the message from A to B and substitutes D's identity for A's. The consequence is that B believes he is sharing the key with D whereas he is in fact sharing it with A. Maybe B will give D's confidential information to A.

Security Assumption 3

The adversary may be a legitimate protocol participant (an insider), or an external party (an outsider), or a combination of both.

This assumption leads to an alternative attack against the second protocol:

Second Attempt III

1. $A \longrightarrow C$: A, B
2. $C \longrightarrow S$: A, C
3. $S \longrightarrow C$: $\{K_{AC}\}_{K_{AS}}, \{K_{AC}\}_{K_{CS}}$
4. $C \longrightarrow A$: $\{K_{AC}\}_{K_{AS}}, \{K_{AC}\}_{K_{CS}}$
5. $A \longrightarrow C$: $\{K_{AC}\}_{K_{CS}}, A$

Now A thinks he is communicating with B, but in reality he communicates with C who is able to read all the messages sent by A.

Third Attempt I

The previous attacks show that we must add the identities of the participants into the messages in a secure way. This leads to the following attempt:

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_{AB,B}\}_{K_{AS}}, \{K_{AB}, A\}_{K_{BS}}$
3. $A \longrightarrow B$: $\{K_{AB}, A\}_{K_{BS}}$

However, even this version is not completely satisfactory.

Security Assumption 4:
An adversary is able to obtain the value of the session key K_{AB} used in any sufficiently old previous run of the protocol.

The attack based on this assumption as follows:

Third Attempt II

1. $A \longrightarrow C$: A, B
2. $C \longrightarrow A$: $\{K'_{AB,B}\}_{K_{AS}}, \{K'_{AB}, A\}_{K_{BS}}$
3. $A \longrightarrow B$: $\{K'_{AB}, A\}_{K_{BS}}$

This time C intercepts the message from A to S. The key K'_{AB} is an old session key used by A and B in a previous session. Because K'_{AB} is old, C has maybe succeeded to break it. Even if K'_{AB} has not been broken, C could replay old messages in this new session, and it can cause a lot of problems.

Fourth Attempt

In order to prevent replays, we must add timestamps or nonces into the protocol. A nonce is a random value generated by one party and returned to that party to show that a message is newly generated.
The fourth version uses nonces:

1. $A \longrightarrow S$: A, B, N_A
2. $S \longrightarrow A$: $\{K_{AB}, B, N_A, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \longrightarrow B$: $\{K_{AB}, A\}_{K_{BS}}$
4. $B \longrightarrow A$: $\{N_B\}_{K_{AB}}$
5. $A \longrightarrow B$: $\{N_B - 1\}_{K_{AB}}$

This protocol is the famous protocol of Needham and Schroeder, published in 1978. Unfortunately, it still has a flaw. If an attacker knows old session keys, he can use such in the last three messages. Then B thinks he is communicating with A using a new session key while actually he is communicating with the attacker using an old session key.

Fifth and Final Attempt

By adding nonces we achieve finally a correct protocol:

1. $A \longrightarrow S$: A, B, N_A, N_B
2. $S \longrightarrow A$: $\{K_{AB}, B, N_A\}_{K_{AS}}, \{K_{AB}, A, N_B\}_{K_{BS}}$
3. $A \longrightarrow B$: $\{K_{AB}, A, N_B\}_{K_{BS}}$
4. $B \longrightarrow A$: B, N_B

Concepts Related to Key Establishment Protocols I

Definition

A key transport protocol is a key establishment protocol in which one of the principals generates the key and this key is then transferred to all protocol users.

Definition

A key agreement protocol is a key establishment protocol in which the session key is a function of inputs by all protocol users.

Concepts Related to Key Establishment Protocols II

Definition

A hybrid protocol is a key establishment protocol in which the session key is a function of inputs by more than one principal, but not by all users. This means that the protocol is a key agreement protocol from the viewpoint of some users, and a key transport protocol from the viewpoint of others.

Key establishment protocols pay usually attention on compromised keys. Especially, the following properties are important.

Definition

A key establishment protocol provides forward secrecy if compromise of the long-term keys of a set of principals does not compromise the session keys established in previous protocols runs involving those principals.

Concepts Related to Key Establishment Protocols III

Definition

A protocol provides partial forward secrecy if compromise of the long-term keys of one or more specific principals does not compromise the session keys established in previous protocols runs involving those principals.

Definition

A protocol provides resistance to key compromise impersonation if compromise of a long-term key of a principal A does not allow the adversary to masquerade to A as a different principal.

Types of Attacks

- The following list is not complete. The ways in which the adversary may interact with one or more protocol runs are infinite.
- Before condemning new protocols we must remember that different protocols have different objectives. For example, some protocols may have no material to convey confidentially, being concerned solely with real-time authentication.
- Similarly, some protocols may use only light measures against adversaries, because efficiency requirements prevent heavy cryptography or many messages.
- Moreover, one should always clearly define the goals a security protocol is aimed to satisfy. Without such definitions, security proof and analyses are difficult.

Eavesdropping

- Eavesdropping is perhaps the most basic attack on a protocol.
- It is obvious that encryption must be used to protect confidental information such as session keys. In certain protocols there may be other information that also needs to be protected.
- An interesting example is that protocols for key establishment in mobile communications usually demand that the identity of the mobile station remains confidental.
- Eavesdropping is sometimes distinguished as being a passive attack. The other attacks we consider all require the adversary to be active.

Modification

- If any protocol message field is not redundant then modification of it is a potential attack.
- Use of cryptographic integrity mechanisms is therefore pervasive in protocols for authentication and key establishment.
- Whole messages, as well as individual fields, are vulnerable to modification.
- Many attacks do not alter any known message fields at all, but split and re-assemble fields from different messages.
- This means the integrity measures must cover all parts of the message that must be kepttogether; encryption is not enough.

Replay

- Replay attacks include any situation where the adversary interferes with a protocol run by insertion of a message, or part of a message, that has been sent previously in any protocol run.
- We may regard replay as another fundamental type of attack which is often used in combination with other attack elements.
- All security protocols must take this attack into account.
- It is repelled usually by using enumeration of packets, nonces or timestamps.

Preplay

Preplay might be regarded as a natural extension of replay, although it is not clear that this is really an attack that can be useful on its own.

Reflection I

- Reflection is really an important special case of replay.
- A typical scenario is where two principals engage in a shared key protocol and one simply returns a challenge that is intended for itself.
- This attack may only be possible if parallel runs of the same protocol are allowed but this is often a realistic assumption. That is why we pose

Security Assumption 5:
The adversary may start any number of parallel protocol runs between any principals including different runs involving the same principals and with principals taking the same or different protocol roles.

The following example is basic. Suppose A and B already share a secret key K and choose respective nonces N_A and N_B for use in the protocol. The protocol is intended to mutually authenticate both parties by demonstrating knowledge of K.

Reflection II

$$\begin{array}{lll}
1. & A \longrightarrow B: & \{N_A\}_K \\
2. & B \longrightarrow A: & \{N_B\}_K,\ N_A \\
3. & A \longrightarrow B: & N_B
\end{array}$$

On receipt of message 2, A deduces that it must have been sent by B since only B has K. However, if A is willing to engage in parallel protocol runs then there is another possibility, namely that message 2 was originally formed by A. An adversary C can masquerade as B and successfully complete two runs of the protocol:

$$\begin{array}{lll}
1. & A \longrightarrow C_B: & \{N_A\}_K \\
1'. & C_B \longrightarrow A: & \{N_A\}_K \\
2'. & A \longrightarrow C_B: & \{N'_A\}_K,\ N_A \\
2. & C_B \longrightarrow A: & \{N'_A\}_K,\ N_A \\
3. & A \longrightarrow C_B: & N'_A \\
3'. & C_B \longrightarrow A: & N'_A
\end{array}$$

Typing Attacks I

- When a protocol is written on the page its elements are clearly distinct. But in practice a principal receiving a message, whether encrypted or not, simply sees a string of bits which have to be interpreted.
- Typing attacks exploit this by making a recipient misinterpret a message, accepting one protocol element as another one (that is, a message element of a different type). For example, an element which was intended as a principal identifier coould be accepted as a key. Such attacks typically work with replay of a previous message.
- An example can be seen in the well-known protocol of Otway and Rees. A and B share a long-term keys, K_{AS} and K_{BS} respectively, with the server S. S generates a new session key K_{AB} and passes it to B and via B to A. $N_A^{(1)}$ and $N_A^{(2)}$ are nonces chosen by A and N_B is a nonce chosen by B.

Typing Attacks II

1. $A \longrightarrow B$: $N_A^{(1)}, A, B, \{N_A^{(2)}, N_A^{(1)}, A, B\}_{K_{AS}}$
2. $B \longrightarrow S$: $N_A^{(1)}, A, B, \{N_A^{(2)}, N_A^{(1)}, A, B\}_{K_{AS}}, \{N_B, N_A^{(1)}, A, B\}_{K_{BS}}$
3. $S \longrightarrow B$: $N_A^{(1)}, \{N_A^{(2)}, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
4. $B \longrightarrow A$: $N_A^{(1)}, \{N_A^{(2)}, K_{AB}\}_{K_{AS}}$

Typing Attacks III

- The typing attack works because of the similarity in the encrypted parts of the first and last messages – they start with the same message field and are encrypted with the same key.
- As usual for this kind of attack we need to make some extra assumptions if the attack is to succeed.
- The attack depends on the length of the composite field $N_A^{(1)}, A, B$ being the same as that expected for the key K_{AB}.
- This may be quite reasonable assumption: $N_A^{(1)}$ may be 64 bits, A and B could be 32 bits, so that K_{AB} would have to be of length 128 bits, which is a popular choice of symmetric key size today.
- With these assumptions, an adversary C is able to execute an attack. The notation C_B means that the adversary C is masquerading as principal B.

Typing Attacks IV

$$\begin{array}{lll} 1. & A \longrightarrow C_B: & N_A^{(1)},\ A,\ B,\ \{N_A^{(2)}, N_A^{(1)}, A, B\}_{K_{AS}} \\ 4. & C_B \longrightarrow A: & N_A^{(1)},\ \{N_A^{(2)}, N_A^{(1)}, A, B\}_{K_{AS}} \end{array}$$

Typing Attacks V

- C masquerades as B and intercepts the message from A. C then returns the encrypted part of this message to A, which is interpreted by A as message 4 of the protocol.
- A will accept the composite field $N_A^{(1)}, A, B$ as the shared key K_{AB}.
- Of course C knows the values of $N_A^{(1)}$, A and B from message 1, and so is able to continue masquerading as B for the duration of the session.
- Typing attacks can be countered by various methods. Ad hoc precautions include changing the order of the message elements each time they are used, and ensuring that each encryption key is used only once.
- More systematic methods are to include an authenticated message number in each message or an autenticated type field with each field. Naturally these come at a cost in computation and bandwidth.

Cryptanalysis

- Cryptographic algorithms used in protocols are often treated abstractly and considered immune to cryptanalysis. However, there are some exceptions that should be mentioned.
- The most important exception is when it is known that a key is weak and relatively easy to guess once sufficient evidence is available. This evidence will normally be a pair of values, one of which is a function of the key; examples are a plaintext valueand its MAC (message authentication code).
- The most common example of use of a weak key is when the key is formed from a password that needs to be remembered by a human. In this situation the effective key length can be estimated from the set of values that are parctically use as passwords, and is certainly much smaller than would be acceptable as the key length of any modern cryptosystem.

Certificate Manipulation I

- Principals, who make use of a certicate are trusting that a certificate authority has correctly identified the owner of the public key at the time that the certificate was issued.
- However, it is not necessarily expected that the authority is provided with evidence that the correspondending private key is actually held by the principal claiming ownership of the key pair. This leads to potential attacks in which the adversary gains a certificate that a public key is its own, even though it does not know the corresponding private key.
- Consider a key agreement protocol of Matsumoto et al. Principals A and B possess public keys g^a and g^b, respectively, and corresponding private keys a and b.
- Here g generates a suitable group in which the discrete logarithm problem is hard.

Certificate Manipulation II

- Each public key is certified and so A and B possess certificates *Cert(A)* and *Cert(B)* which contain copies of their public keys.

$$\begin{array}{lll} 1. & A \longrightarrow B: & g^x,\ Cert(A) \\ 2. & B \longrightarrow A: & g^y,\ Cert(B) \end{array}$$

- The shared key is $K_{AB} = g^{ay+bx}$, calculated by A as $(g^y)^a(g^b)^x$ and by B as $(g^a)^y(g^x)^b$.
- The adversry C engineers an attack by choosing a random value c, claiming that g^{ac} is its public key, and obtaining a certificate for this public key.
- Notice that C cannot obtain the corresponding private key ac.
- C then masquerades as B, and completes two runs of the protocol, one with A and one with B, as shown in the attack below.

Certificate Manipulation III

$$\begin{array}{lll}
1. & A \longrightarrow C_B: & g^x,\ Cert(A) \\
1'. & C \longrightarrow B: & g^x,\ Cert(C) \\
2'. & B \longrightarrow C: & g^y,\ Cert(B) \\
2. & C_B \longrightarrow A: & g^{yc},\ Cert(B)
\end{array}$$

- After the attacking run is complete, A will calculate the key $K_{AB} = (g^{yc})^a (g^b)^x = g^{acy+bx}$ and B will calculate the key $K_{CB} = (g^{ac})^c (g^x)^b = g^{acy+bx}$.
- Thus A and B have found the same key, but A believes that this key is known only to A and B while B believes it is known only to C and B. This misunderstanding can lead to problems in the subsequent use of the session key.
- Attacks of this sort can be avoided by demanding that every principal demonstrates knowledge of the private key before a certificate is issued for any public key. Such a demonstration is ideally achieved using zero knowledge techniques.

Certificate Manipulation IV

- A more convenient method may be to have the private key owner sign a specific message or challenge.

Protocol Interaction

- Most long-term keys are intended to be used for a single protocol. However, it could be the case that keys are used in multiple protocols. This could be due to careless design, but may be deliberate in cases where devices with small storage capability are used for multiple applications.
- It is easy to see that protocols designed independently may interact badly. For example, a protocol that uses decryption to prove possession of an authenticating key may be used by an adversary to decrypt messages from another protocol, if the same key is used.
- One method to prevent such attacks is to include the protocol details (such as unique identifier and version number) in an authenticated part of the protocol messages.

Design Principles for Cryptographic Protocols I

Adadi and Needham have proposed a set of principles intended to act as 'rules of thumb' for protocol designers. They were derived from observation of the most common errors that have been found in published protocols. Of course, new kind of errors or attacks will be found, but the following list helps at least to avoid old mistakes.

1. Every message should say what it means: the interpretation of the message should depend only on its content.
2. The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.
3. If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.
4. Be clear about why encryption is being done.

Design Principles for Cryptographic Protocols II

5. When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message.
6. Be clear about what properties you are assuming about nonces.
7. If a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay a response.
8. If timestamps are used as freshness guarantees, then the difference between local clocks at various machines must be much less than the allowable age of a message.
9. A key may have been used recently, for example to encrypt a nonce, and yet be old and possibly compromised.

Design Principles for Cryptographic Protocols III

10. It should be possible to deduce which protocol, and which run of that protocol, a message belongs to, and to know its number in the protocol.
11. The trust relations in a protocol should be explicit and there should be good reasons for the necessity of these relations.

Bellare-Rogaway MAP1 Protocol I

We start with an instructive example of a protocol which has formally been proved correct, but which anyway has a flaw. The protocol, Bellare-Rogaway MAP1, assumes that both entities have the same secret key. The entities try to authenticate each other.
We use the following notations:

- $\{M\}_K$: encryption of message M with key K to provide confidentiality and integrity.
- $[M]_K$: one-way transformation of message M with key K to provide integrity.

The MAP1 mutual authentication protocol was proposed in a landmark paper of Bellare and Rogaway. They provided a formal definition and showed that MAP1 is provably secure.

Bellare-Rogaway MAP1 Protocol II

$$\begin{array}{lll} 1. & A \longrightarrow B: & N_A \\ 2. & B \longrightarrow A: & N_B, [B, A, N_A, N_B]_{K_{AB}} \\ 3. & A \longrightarrow B: & [A, N_B]_{K_{AB}} \end{array}$$

- In the Bellare-Rogaway model of security, an adversary may only interact with sessions of the same protocol.
- An attack of Alves-Foss shows why this assumption is insufficient. He designed the following protocol, known as EVE1, which can also be shown to be provably secure:

$$\begin{array}{lll} 1. & A \longrightarrow B: & N_A \\ 2. & B \longrightarrow A: & N_B, [A, B, N_A, N_B]_{K_{AB}} \\ 3. & A \longrightarrow B: & [A, N_B]_{K_{AB}} \end{array}$$

Bellare-Rogaway MAP1 Protocol III

.

A chosen protocol attack on the MAP1 protocol is now possible. Suppose I is an adversary who wishes to attack the protocol. In the following attack A is used as an oracle against himself. In the attack,

I masquerades as B in a run of the MAP1 protocol started by A.

In parallel, I starts a run of the EVE1 protocol with A while masquerading as B.

$$\begin{array}{lll}
1. & A \longrightarrow I_B: & \mathrm{N}_A \\
1'. & I_B \longrightarrow A: & N_A \\
2'. & A \longrightarrow I_B: & N'_A,\ [B, A, N_A, N'_A]_{K_{AB}} \\
2. & I_B \longrightarrow A: & N'_A,\ [B, A, N_A, N'_A]_{K_{AB}} \\
3. & A \longrightarrow I_B: & [A, N'_A]_{K_{AB}}
\end{array}$$

- Is the attack on the MAP1 protocol valid?

Bellare-Rogaway MAP1 Protocol IV

- A reasonable conclusion may be that the attack is invalid since it violates an assumption of the model used in proving the protocol secure.
- On the other hand, the attack is a reminder that provably security does not guarantee security against a chosen protocol attacks.
- There is also an article which pays attention on security proofs: Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock: Errors in Computational Complexity Proofs for Protocols, *ASIACRYPT 2005, LNCS 3788*, pp. 624-643.

ISO/IEC 9798 -standards I

The international standard ISO/IEC 9798 Part 2 specifies six authentication protocols using symmetric encryption algorithms. Four of these protocols are intended to provide entity authentication alone, while two are intended to provide key establsihment as well as entity authentication.
We present here the fourth protocol:

$$\begin{array}{lll}
1. & B \longrightarrow A: & N_B \\
2. & A \longrightarrow B: & \{N_A, N_B, B\}_{K_{AB}} \\
3. & B \longrightarrow A: & \{N_B, N_A\}_{K_{AB}}
\end{array}$$

No attack against this protocol has been found.

Server-Less Key Establishment

- We present two protocols that allow keys to be established directly between two users without the use of a server.
- The protocol require that the two users share a long-term secret key.
- In what follows, K_{AB} is a long-term secret key shared by the entities and K'_{AB} is a new session key.

Andrew Secure RPC Protocol I

$$\begin{array}{lll}
1. & A \longrightarrow B: & \{N_A\}_{K_{AB}} \\
2. & B \longrightarrow A: & \{N_A + 1, N_B\}_{K_{AB}} \\
3. & A \longrightarrow B: & \{N_B + 1\}_{K_{AB}} \\
4. & B \longrightarrow A: & \{K'_{AB}, N'_B\}_{K_{AB}}
\end{array}$$

- Burrows et al. have pointed out a major problem with the protocol: A has no assurance that K'_{AB} is fresh.
- An intruder could substitute a previously recorded message 4 and force A to accept an old, possibly compromised, session key.
- Clark and Jacob pointed out another problem, using the following attack:

$$\begin{array}{lll}
1. & A \longrightarrow B: & \{N_A\}_{K_{AB}} \\
2. & B \longrightarrow A: & \{N_A + 1, N_B\}_{K_{AB}} \\
3. & A \longrightarrow B: & \{N_B + 1\}_{K_{AB}} \\
4. & I_B \longrightarrow A: & \{N_A + 1, N_B\}_{K_{AB}}
\end{array}$$

Andrew Secure RPC Protocol II

- The result of the attack is that A accepts the value $N_A + 1$ as a session key with B.
- The potential damage of the attack depends on what property the nonce N_A is assumed to have. If N_A is a predictable nonce such as a counter value, then the attacker could force A into accepting a bogus quantity as a session key, whose value could be known to the attacker.
- If N_A were random, however, then the potential damage is not so immediate since there is no release of the session key.

Burrows et al Modification I

Burrows et al. has suggested a modification of Andrew's protocol, where the treatment of the nonces are changed. The nonce N_A needs not to be secret and the nonce N_B could be omitted altogether:

$$\begin{array}{lll}
1. & A \longrightarrow B: & A, N_A \\
2. & B \longrightarrow A: & \{N_A, K'_{AB}\}_{K_{AB}} \\
3. & A \longrightarrow B: & \{N_A\}_{K'_{AB}} \\
4. & B \longrightarrow A: & N'_B
\end{array}$$

Lowe has published the following attack. An intruder engages in two protocol runs with A while masquerading as B. In one of these runs I is the initiator of the protocol, while in the other A is induced to act as initiator.

Burrows et al Modification II

1.	$A \longrightarrow I_B$:	A, N_A
1'.	$I_B \longrightarrow A$:	B, N_A
2'.	$A \longrightarrow I_B$:	$\{N_A, K'_{AB}\}_{K_{AB}}$
2.	$I_B \longrightarrow A$:	$\{N_A, K'_{AB}\}_{K_{AB}}$
3.	$A \longrightarrow I_B$:	$\{N_A\}_{K'_{AB}}$
3'.	$I_B \longrightarrow A$:	$\{N_A\}_{K'_{AB}}$
4.	$I_B \longrightarrow A$:	N_I
4'.	$A \longrightarrow I_B$:	N'_A

The result of the attack is that A has completed two successful runs, apparently with B, although B has not engaged in the protocol.

Boyd's protocol

Boyd has invented a surprisingly simple protocol which seems to be correct.

$$\begin{array}{lll} 1. & A \longrightarrow B: & N_A \\ 2. & B \longrightarrow A: & N_B \end{array}$$

- The new key is $K'_{AB} = f(N_A, N_B, K_{AB})$.
- Function f must be such that it infeasible to find the value of f without knowing K_{AB}.
- Typically, f could be a MAC function.

Server-Based Key Establishment

We use the following notations:

- A and B are two users wishing to establish a session key.
- S is the server.
- K_{AS} and K_{BS} are long-term keys, initially shared by A and S, and by B and S.
- K_{AB} is a session key shared by A and B.

Denning-Sacco Protocol I

We have already seen Needham-Schroeder protocol which contains a flaw. Denning and Sacco suggested the following protocol as a solution:

$$\begin{array}{lll}
1. & A \longrightarrow S: & A, B \\
2. & S \longrightarrow A: & \{B, K_{AB}, T_S, \{A, K_{AB}, T_S\}_{K_{BS}}\}_{K_{AS}} \\
3. & A \longrightarrow B: & \{A, K_{AB}, T_S\}_{K_{BS}}
\end{array}$$

- Here T is a timestamp. When B receives the message in step 3, he rejects it, if the timestamp is too old.
- This method demands synchronization of clocks in a distributed system which may be not simple.
- There are a lot of other protocols which do not use timestamps. Maybe the best known is Otway-Rees Protocol, but it contains also a flaw. We skip these protocols.

Robust Principles for Public Key Protocols I

Anderson and Needham have proposed a set of what they call robustness principles for publi-key-based protocols. These can be considered as more specific instances of the general principles for protocol design proposed earlier by Adabi and Needham.

1. Sign before encrypting. If a signature is affixed to encrypted data then one cannot assume that the signer has any knowledge of the data.
2. Be careful how entities are distinguished. If possible avoid using the same key for two different purposes (such as signing and decryption) and be sure to to distinguish different runs of the same protocol from each other.
3. Be careful when signing or decrypting data that you never let yourself be used as an oracle by your opponent.
4. Account all the bits: how many provide equivocation, redundancy, computational complexity, and so on.

Robust Principles for Public Key Protocols II

5. Do not assume that a message you receive has a particular form (such as g^r for known r) unless you can check this.
6. Do not assume the secrecy of anybody else's 'secrets' (except possibly those of a certification authority).
7. Be explicit about the security parameters of cryptographic primitives.

Syverson (Limitations on design principles for public key protocols) questioned the applicability of some of the principles by showing examples when they are not appropriate. Nevertheless, Syverson concludes that the principles are still useful but should be used intelligently and critically by the protocol designer. Especially, the first principle is often ignored.

ISO/IEC 11770-3 -protocols I

The standard specifies six key transport protocols in a generic fashion and with some optional items. We study three of these.
The first is very simple and shows the power of public key cryptography:

$$1. \quad \mathrm{A} \longrightarrow \mathrm{B}: \quad E_B(A, K_{AB}, T_A)$$

where E_B is encryption function using B's public key.

- It is essential to assume that the public key encryption used provides non-malleability
  An encryption scheme provides non-malleability, if it is infeasible to take an existing ciphertext and transfer it into a related ciphertext without knowledge of the input plaintext.
- If the protocol does not satisfy non-malleability, then the adversary may be able to change the value of the fields A and T_A included in the encrypted message.

ISO/IEC 11770-3 -protocols II

- However, the standard does not mention this property.
- The protocol works well with respect to A.
- On the other hand, A achieves no assurance with regard to key confirmation, or even that B is operative.
- Moreover, B cannot be sure with whom he is communicating.
- The timestamp T_A guarantees the freshness of the message, but not the freshness of the key. That is why the timestamp is optional in the standard.

In mechanism 2, authentication is added:

1. A $\longrightarrow$ B: $B, T_A, E_B(A, K_{AB}), Sig_A(B, T_A, E_B(A, K_{AB}))$

- Again, no key confirmation from B, but the signature of A allows B to achieve key confirmation.

ISO/IEC 11770-3 -protocols III

- This protocol violates the first principle of Andersson and Needham, but this does not seem to lead into difficulties.

The final protocol 6 achieves mutual entity authentication and mutual key confirmation.

$$\begin{array}{lll} 1. & A \longrightarrow B: & E_B(A, K_{AB}, N_A) \\ 2. & B \longrightarrow A: & E_A(B, K_{BA}, N_A, N_B) \\ 3. & A \longrightarrow B: & N_B \end{array}$$

No attacks are known against this protocol.

Needham-Schroeder Public Key Protocol I

The Needham-Schroeder public key protocol was one the earliest published key establishment protocols along with it well-known companion using symmetric encryption.

$$\begin{array}{lll} 1. & A \longrightarrow B: & E_B(N_A, A) \\ 2. & B \longrightarrow A: & E_A(N_A, N_B) \\ 3. & A \longrightarrow B: & E_B(N_B) \end{array}$$

In 1996, an attack was found with the help of the automatic analysing tool FDR:

$$\begin{array}{lll} 1. & A \longrightarrow C: & E_C(N_A, A) \\ 1'. & C_A \longrightarrow B: & E_B(N_A, A) \\ 2'. & B \longrightarrow C_A: & E_A(N_A, N_B) \\ 2. & C \longrightarrow A: & E_A(N_A, N_B) \\ 3. & A \longrightarrow C: & E_C(N_B) \\ 3'. & C_A \longrightarrow B: & E_B(N_B) \end{array}$$

Needham-Schroeder Public Key Protocol II

The correction is simple:

$$\begin{array}{lll} 1. & A \longrightarrow B: & E_B(N_A, A) \\ 2. & B \longrightarrow A: & E_A(N_A, N_B, B) \\ 3. & A \longrightarrow B: & E_B(N_B) \end{array}$$

Key Generation with Diffie-Hellman

- So far we have studied protocols which transport keys from one entity to another.
- The other possibility is to generate keys together, without transporting them at all.
- These methods are based on the Diffie-Hellman key generating method. The basic method suffers from man-in-the-middle attack, so something must be added to the basic scheme.
- There are many suggestions: MTI protocols, MQV, STS, Oakley, SKEME, IKE etc.
- We present Station-to-Station (STS) protocol, because it is simple and its modified version seems correct.
- We study also IKE which is the key agreement phase of the IPSec protocol.

Station-to-Station Protocol I

In the protocol, A and B use $\mathbf{Z}_q$ with q prime and with a generator (primitive root) g. A and B calculate $t_A = g^{r_A}$, $t_B = g^{r_B}$, respectively, where r_A and r_B are random numbers. Then:

$$\begin{array}{lll}
1. & A \longrightarrow B: & A, B, t_A \\
2. & B \longrightarrow A: & B, A, t_B, \{\mathrm{Sig}_B(t_B, t_A)\}_{K_{AB}} \\
3. & A \longrightarrow B: & A, B, \{\mathrm{Sig}_A(t_A, t_B)\}_{K_{AB}}
\end{array}$$

where $K_{AB} = g^{r_A r_B}$ (or derived from $g^{r_A r_B}$).
This version seems to prevent the man-in-the-middle attack. There is, however, a similar attack:

$$\begin{array}{lll}
1. & A \longrightarrow C_B: & A, B, t_A \\
1'. & C \longrightarrow B: & C, B, t_A \\
2'. & B \longrightarrow C: & B, C, t_B, \{\mathrm{Sig}_B(t_B, t_A)\}_{K_{AB}} \\
2. & C_B \longrightarrow A: & B, A, t_B, \{\mathrm{Sig}_B(t_B, t_A)\}_{K_{AB}} \\
3. & A \longrightarrow C_B: & A, B, \{\mathrm{Sig}_A(t_A, t_B)\}_{K_{AB}}
\end{array}$$

Station-to-Station Protocol II

Result: B has no indication that A has engaged in the protocol and yet A has completed a succesful run and accepted that her partner is B.
These kind of unknown key-share attacks can be prevented by including the name of the partner entity in the signatures exchanged. Moreover, this change provides an explicit indication of the peer entity so that the stronger form of entity authentication is achieved.
In addition, there no longer seems to be a need for the symmetric encryption:

$$\begin{array}{lll} 1. & A \longrightarrow B: & t_A \\ 2. & B \longrightarrow A: & t_B, \mathrm{Sig}_B(t_B, t_A, A) \\ 3. & A \longrightarrow B: & \mathrm{Sig}_A(t_A, t_B, B)\} \end{array}$$

Internet Key Exchange, IKE I

- IKE is the key establishment protocol of IPSec.
- IPSec is used to communicate confidentally over a public network. By creating a so-called security association IPSec transports encrypted and authenticated packets. It is a network layer protocol and it is used to build virtual private networks.
- Moreover, if IPSec is applied between gateways or routers, its tunnel mode hides the source and destination IP-addresses.
- In order to create and to use a security association, symmetric session keys must be established first.
- Many protocols have been suggested for the key establishment of IPSec: Oakley, SKEME, Photuris, ISAKMP/Oakley. After many, confusing, years, a single Internet standard emerged in 1998, known as Internet Key Exchange or IKE.

Internet Key Exchange, IKE II

- However, the first versions of IKE were complicated and they generated a lot of critisism. For example, one early version had eight different modes of operation.
- The newest version, IKEv2, developed 2004-05, has only one operation mode with some options. We describe here the option which uses public key cryptography with certificates.
- Our presentation concentrates on the security aspects of IKE, so we drop the headers from the messages. Typically, headers contain version numbers and flags of various sort plus security parameter index SPI which identifies the security association uniquely.
- Similarly, we drop SA-payloads which are used to negotiate cryptographic algorithms for encryption, digital signatures and hash functions.

The algorithm:

Internet Key Exchange, IKE III

$$\begin{array}{lll}
1. & A \longrightarrow B: & t_A, N_A \\
2. & B \longrightarrow A: & t_B, N_B, C_B \\
3. & A \longrightarrow B: & \{A, C_A, \mathrm{Auth}\}_{K_{AB}} \\
4. & B \longrightarrow A: & \{B, C_B, \mathrm{Auth}\}_{K_{BA}}
\end{array}$$

Here:

1. t_A and t_B are the Diffie-Hellman values calculated by A and B, respectively.
2. N_A and N_B are nonces.
3. C_B and C_A are the public key certificates of B and A, respectively.
4. Auth is a block of data signed by the sender's secret key.
5. The keys K_{AB} and K_{BA} are different, but they have been derived from the common DH value. So both A and B know these keys.

Internet Key Exchange, IKE IV

- An expected attack against IKE is state and CPU exhaustion, where the target is flooded with session initiation requests from forged IP addresses.
- Against this attack, IKE has a possibility to use cookies in the first response message. A cookie contains a challenge which must be solved and sent back by the initiator.
- The first three messages look in this case as follows:

$$\begin{array}{lll} 1. & A \longrightarrow B: & t_A, N_A \\ 2. & B \longrightarrow A: & \text{Cookie} \\ 3. & A \longrightarrow B: & \text{Cookie},\ t_A, N_A \end{array}$$

Internet Key Exchange, IKE V

After this, the protocol continues as before.

B should send a cookie back when it detects a large number of half-finished IKE-negotiations.

There are many other details which must be taken into account in this type of a practical protocol. For example:

- Retransmission timers.
- Use of sequence numbers (against reply).
- Window size for overlapping requests.
- State synchronization and connection timeouts.
- Rekeying.
- Reuse of DH-exponentials.
- Generating keying material.
- Extensible authentication methods (EAP).
- Error handling.
- NAT traversal.