



安全协议设计与分析

第四讲：攻击与设计原则

李晖 网络空间安全学院

lihuill@bupt.edu.cn



last Lecture: Goals for security protocol



- To know if a protocol is secure you must know what it aims to achieve.
- Example: Diffie-Hellman & STS Protocol.

Today: Attacks and Principles

- Common types of attacks on protocols.
- Good design principles for protocol.

Some Common Types of Attack



- Eavesdropping
- Modification
- Replay / Preplay
- Reflection
- Man-in-the-Middle
- Denial of Service
- Typing Attack
- Parallel Session Attack

Eavesdropping



- An Eavesdropping attack only passively observe messages.
- Protocols defend against Eavesdropping attacks by using encryption for confidentiality.
- The attacker is a **passive outsider**.

Modification



- A Modification attack alters or replaces some messages.
- defend against Modification attacks by:
 - **Encryption**
 - **MAC**
 - **Hash with secret key**
 - **Signature (Non-optimal option)**

Replay / Preplay



- The attacker sends a message that it has observed as part of the protocol run.
- Protocols defend against replay attacks by :
 - **Challenge-Response**
 - **Timestamp**
 - **Sequence Number**

Reflection

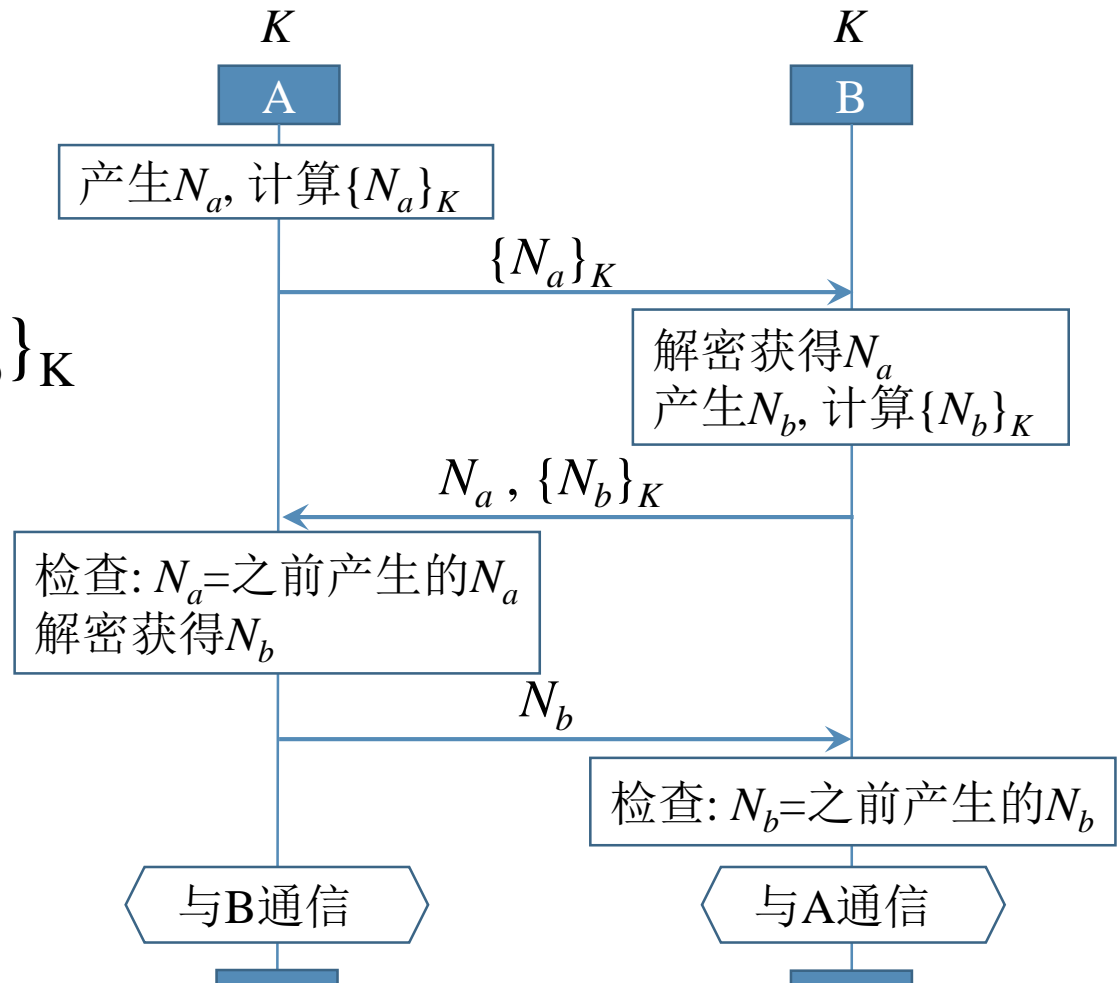


- Reflection attacks are a kind of **replay attack** that use a protocol against itself.
- The attacker provides the “proof” of authentication by challenging the challenger.

Reflection Attack Example

In this protocol A and B share the key K . They want to ensure they both take part in the protocol.

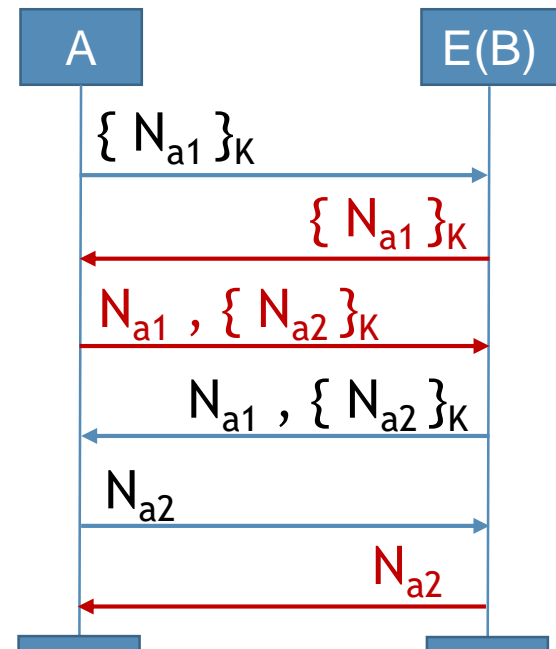
1. $A \rightarrow B : \{N_a\}_K$
2. $B \rightarrow A : N_a, \{N_b\}_K$
3. $A \rightarrow B : N_b$



Reflection Attack Example

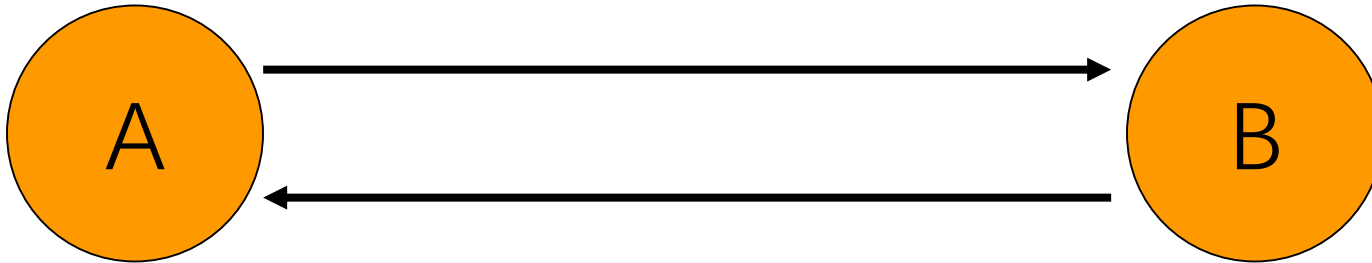


1. $A \rightarrow E(B) : \{ N_{a1} \}_K$
1' . $E(B) \rightarrow A : \{ N_{a1} \}_K$
2' . $A \rightarrow E(B) : N_{a1} , \{ N_{a2} \}_K$
2. $E(B) \rightarrow A : N_{a1} , \{ N_{a2} \}_K$
3. $A \rightarrow E(B) : N_{a2}$
3' . $E(B) \rightarrow A : N_{a2}$



Man-in-the-Middle

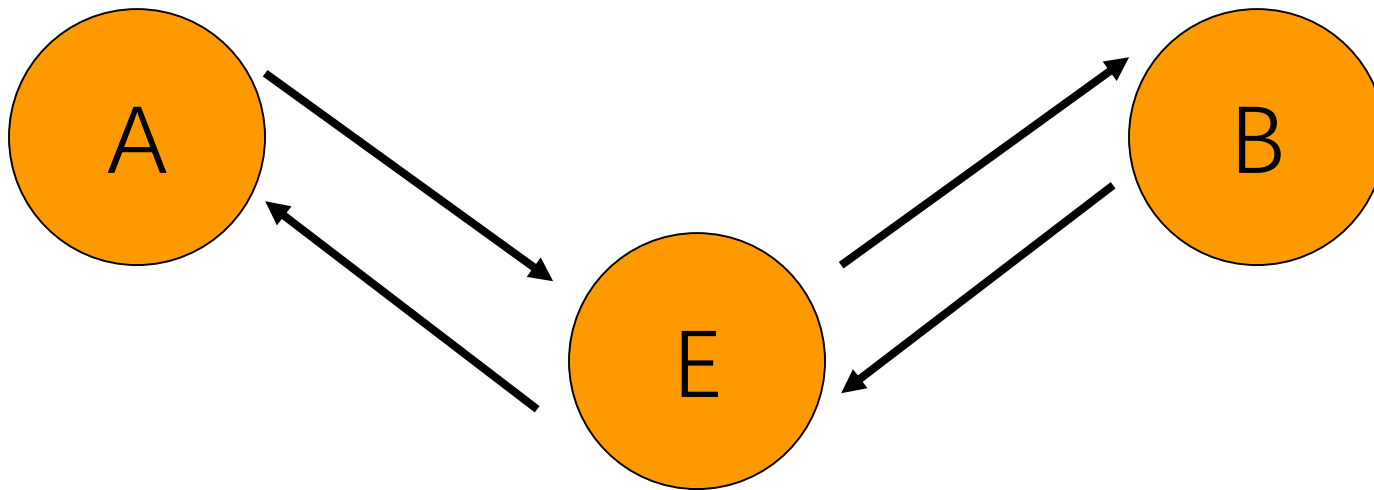
- In a **Man-in-the-Middle attack**, the attacker gets in the middle of a real run of a protocol.



Man-in-the-Middle



- In a **Man-in-the-Middle attack** the attacker gets in the middle of a real run of a protocol.



- defend against MITM attacks by:
 - **Authentication**
 - **Adding pre-shared secret information between entities**

Denial of Service (DoS)



- Every communication request uses an amount of memory and CPU.
- A DoS attack tries to use up all of a sever's CPU or memory by making 1,000,000s of requests.
- All systems can be subject to a DoS attack...
... but some protocols can make this better or worse.

A Protocol Vulnerable to Denial of Service



A uses its public key K_a to establish a session key K_{as}

1. $A \rightarrow S : A, N_a$
2. $S \rightarrow A : E_{K_a} (N_a, N_s, K_{as})$
3. $A \rightarrow S : \{ N_s \}_{K_{as}}$

S is particularly vulnerable to a DoS attack because for each connection S has to:

- generate a nonce and a key,
- perform a public key encryption,
- allocate memory for the nonce and the key.

A Protocol Resistant to Denial of Service



A uses S's public key K_s to establish a session key K_{as}

1. $A \rightarrow S : E_{K_s}(A, S, N_a, K_{as})$
2. $S \rightarrow A : \{ N_a \}_{K_{as}}$

Now A has to do the expensive encryption in order to make S do any more than a single decryption.

Therefore may more “bots” would be needed for a successful attack.

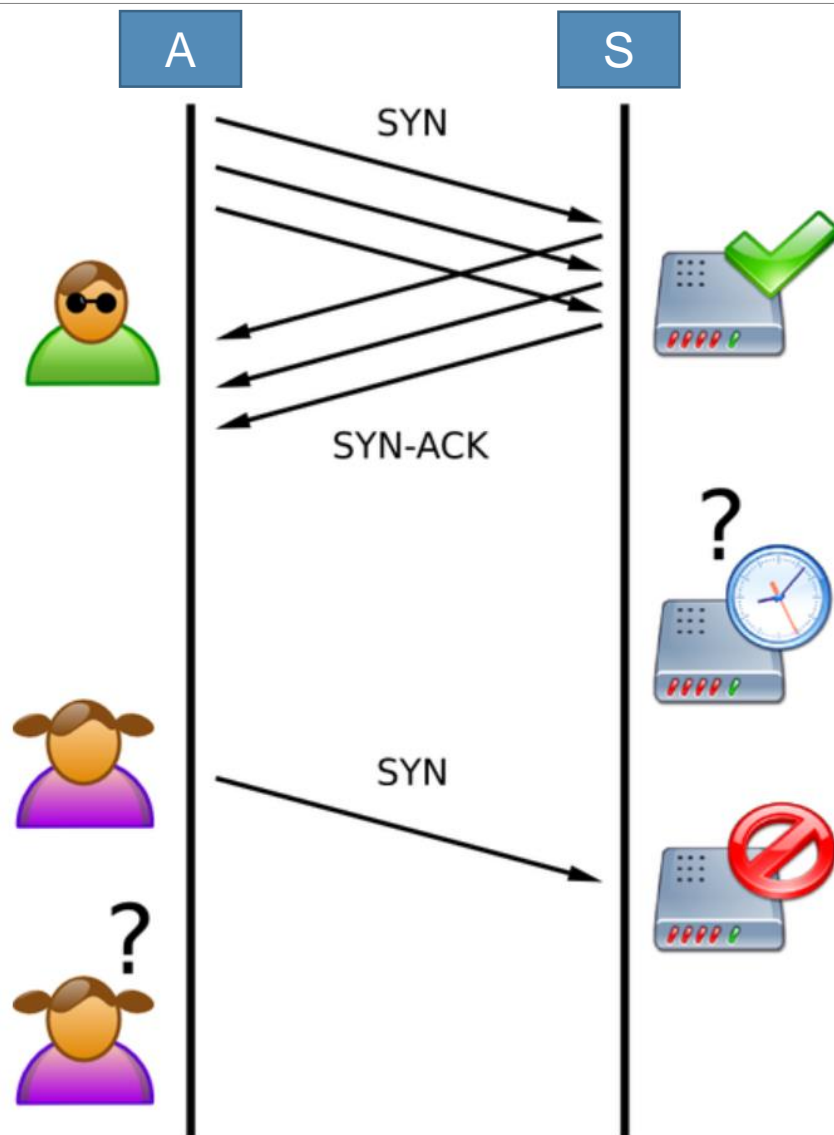
SYN flood DoS Attack



TCP starts a session by:

1. $A \rightarrow S : \text{SYN}$
 2. $S \rightarrow A : \text{ACK, SYN}$ (add A to the table of connections)
 3. $A \rightarrow S : \text{ACK}$ (~ 3 min. time out)
- The “SYN flood” attack sends lots of SYN messages to S and fills its tables, therefore real requests will be ignored.

SYN flood DoS Attack



Typing Attack



- In a typing attack the attacker passes off one type of message as being another.
- This kind of attack may not work on a real implementation...
 - ... but is also hard to spot.

Typing Attack Example



- **Andrew secure RPC protocol** is a handshake, then a key distribution:

1. $A \rightarrow B : \{ N_A \}_{K_{ab}}$
2. $B \rightarrow A : \{ N_A + 1, N_B \}_{K_{ab}}$
3. $A \rightarrow B : \{ N_B + 1 \}_{K_{ab}}$
4. $B \rightarrow A : \{ K_s, N' \}_{K_{ab}}$



Typing Attack Example

1. $A \rightarrow B : \{ N_A \}_{K_{ab}}$
2. $B \rightarrow A : \{ N_A + 1, N_B \}_{K_{ab}}$
3. $A \rightarrow B : \{ N_B + 1 \}_{K_{ab}}$
4. $E(B) \rightarrow A : \{ N_A + 1, N_B \}_{K_{ab}}$

- The attacker replays message 2. "A" now uses the wrong key...
- but the attacker only learns it if N_A is predictable.

Parallel Session Attack



- the attacker can use the messages in one session to synthesize the messages in another session, since the protocol can be executed in parallel.
- The attacker can obtain the required information by exchanging appropriate protocol messages

Parallel Session Attack Example



- Woo-Lam protocol (**Goal: B authenticate A**)

1. $A \rightarrow B : A$

2. $B \rightarrow A : N_B$

3. $A \rightarrow B : E_{K_{AS}}(N_B)$

4. $B \rightarrow S : E_{K_{BS}}(A, E_{K_{AS}}(N_B))$

5. $S \rightarrow B : E_{K_{BS}}(N_B)$

Parallel Session Attack Example



- Woo-Lam protocol (**Goal: B authenticate A**)

1. $A(E) \rightarrow B : A$

I. $E \rightarrow B : E$

2. $B \rightarrow A(E) : N_B$

II. $B \rightarrow E : N_B'$

3. $A(E) \rightarrow B : E_{KES}(N_B)$

III. $E \rightarrow B : E_{KES}(N_B)$

***not N_B'

4. $B \rightarrow S : E_{KBS}(A, E_{KES}(N_B))$

***S cannot get N_B

IV. $B \rightarrow S : E_{KBS}(E, E_{KES}(N_B))$

***S can decrypt correctly

5. $S \rightarrow B : E_{KBS}(\dots)$

V. $S \rightarrow B : E_{KBS}(N_B)$

Some Common Types of Attack



- Eavesdropping
- Modification
- Replay / Preplay
- Reflection
- Man-in-the-Middle
- Denial of Service
- Typing Attack
- Parallel Session Attack

Good Protocol Design

- The best way to avoid protocol faults is to design them right in the first place.
- Suggested by Martin Abadi and Roger Needham



Martín Abadi



Roger Needham
(1935-2003)

Roger Needham's theoretical contributions :

- development of **BAN** logic ;
- His **NS** protocol (co-invented with Michael Schroeder) forms the basis of the Kerberos authentication and key exchange system.
- co-designed the TEA and XTEA encryption algorithms.
- He pioneered the technique of protecting passwords using a one-way hash function.

Principle 0 (高效原则)



- The protocol must be efficient
 - No unnecessary encryption
 - Don't include message you don't need.
- Main concerns about efficiency
 - Computational efficiency
 - encryption and decryption functions
 - generation and verification of digital signatures
 - message authentication codes
 - calculation of hash functions
 - Communications efficiency
 - length of messages and number of rounds
 - Storage efficiency
- Problem: Principle 0 goes against most of the other principles.

Example: Kerberos Update



An old version two of Kerberos ran as follows:

1. $A \rightarrow S : A, B, N_A$
2. $S \rightarrow A : \{K_{AB}, B, L, N_A, \{K_{AB}, A, L\}_{K_{BS}}\}_{K_{AS}}$
3. $A \rightarrow B : \{A, T_A\}_{K_{AB}}, \{K_{AB}, A, L\}_{K_{BS}}$
4. $B \rightarrow A : \{T_A + 1\}_{K_{AB}}$

N.B. note the use of double encryption in 2.

Example: Kerberos Update



A newer version is:

1. $A \rightarrow S : A, B, N_A$
2. $S \rightarrow A : \{K_{AB}, B, L, N_A\}_{K_{AS}}, \{K_{AB}, A, L\}_{K_{BS}}$
3. $A \rightarrow B : \{A, T_A\}_{K_{AB}}, \{K_{AB}, A, L\}_{K_{BS}}$
4. $B \rightarrow A : \{T_A + 1\}_{K_{AB}}$

Double encryption removed: it's expensive and unnecessary.

Principle 1 (消息独立完整性原则)

- Every message should say what it means: the interpretation of the message should depend only on its content.

It should be possible to write down a straightforward sentence describing what the message means.



Meaning of Messages



For instance the Needham-Schroeder Protocol:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b)$
3. $A \rightarrow B : E_B(N_b)$

Message 1: $E_X(Y, Z)$ means I am Z and I want to communicate with X using Y.

Meaning of Messages



For instance the Needham-Schroeder Protocol:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b)$
3. $A \rightarrow B : E_B(N_b)$

Message 2: $E_X(Y, Z)$ means **someone** wants to communicate X using Y and Z.

Meaning of Messages



For instance the Needham-Schroeder Protocol:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b)$
3. $A \rightarrow B : E_B(N_b)$

$E_A(N_a, N_b)$ does not mean that **B** wants to communicate with A using N_a & N_b because there is no reference to B

Meaning of Messages



The corrected version fixes this:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b, \mathbf{B})$
3. $A \rightarrow B : E_B(N_b)$

Message 2: $E_X(Y, Z, \mathbf{W})$ means **I am W** and I want to communicate X using Y and Z.

Meaning of Messages



For instance the Needham-Schroeder Protocol:

1. $A \rightarrow B : E_B(\mathbf{N}_a, \mathbf{A})$
2. $B \rightarrow A : E_A(N_a, \mathbf{N}_b, \mathbf{B})$
3. $A \rightarrow B : E_{\mathbf{B}}(N_b)$

Message 3: $E_X(Y)$ means someone accepts communication with X using Y.

Here we don't need to mention A because only A knows N_b

Principle 2 (消息前提准确原则)



- 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.
- For example: the entity's public key is known to all others

Principle 3 (主体身份标识原则)

- If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name in the message.



Example of Principle 3



The following protocol lets B authenticate A using a trusted server S

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_b$
3. $A \rightarrow B : \{ N_b \}_{K_{as}}$
4. $B \rightarrow S : \{ A, \{ N_b \}_{K_{as}} \}_{K_{bs}}$
5. $S \rightarrow B : \{ N_b \}_{K_{bs}}$

Example of Principle 3

1. $E(A) \rightarrow B : A$

1' . $E \rightarrow B : E$

2. $B \rightarrow E(A) : N_{ba}$

2' . $B \rightarrow E : N_{be}$

3. $E(A) \rightarrow B : \{ N_{ba} \}_{K_{es}}$

3' . $E \rightarrow B : \{ \mathbf{N}_{ba} \}_{K_{es}}$

4. $B \rightarrow S : \{ A, \{ N_{ba} \}_{K_{es}} \}_{K_{bs}}$

4' . $B \rightarrow S : \{ E, \{ N_{ba} \}_{K_{es}} \}_{K_{bs}}$

5. $S \rightarrow B : \text{Fail}$

5' . $S \rightarrow B : \{ N_{ba} \}_{K_{bs}}$

Example of Principle 3

$$1. \quad E(A) \rightarrow B : A$$

$$1' \quad . \quad E \rightarrow B : E$$

$$2. \quad B \rightarrow E(A) : N_{ba}$$

$$2' \quad . \quad B \rightarrow E : N_{be}$$

$$3. \quad E(A) \rightarrow B : \{ N_{ba} \}_{K_{es}}$$

$$3' \quad . \quad E \rightarrow B : \{ \mathbf{N}_{ba} \}_{K_{es}}$$

$$4. \quad B \rightarrow S : \{ A, \{ N_{ba} \}_{K_{es}} \}_{K_{bs}}$$

$$4' \quad . \quad B \rightarrow S : \{ E, \{ N_{ba} \}_{K_{es}} \}_{K_{bs}}$$

$$5. \quad S \rightarrow B : \{ N_{ba} \}_{K_{bs}}$$

Principle 4 (加密目的原则)



- Be clear about why **encryption** is being done.
- Encryption is not wholly cheap, and not asking precisely why it is being done can lead to redundancy.
- Encryption is not synonymous with security and its improper use can lead to errors.

Principle 5 (签名原则)



- When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message.
- On the other hand, it is proper to infer that the principal that **signs a message then encrypts it** for privacy knows the content of the message.

CCITT X.509

- Was used by a range of governments and banks for public key management.

$A \rightarrow B : A, \text{Sign}_A(T_a, N_a, B, X_a, E_B(Y_a))$

- Supposed to prove that A knows the data X_a , Y_a and keep Y_a secret.
- But **A might not know Y_a**



Principle 6 (随机数的使用原则)

- Be clear what properties you are assuming about nonces.
- What may do for ensuring temporal succession may not do for ensuring association - and perhaps association is best established by other means.
- Example: (Otway-Rees protocol)

message1 $A \rightarrow B : M, A, B, \{N_a, M, A, B\}_{K_{as}}$

message2 $B \rightarrow S : M, A, B, \{N_a, M, A, B\}_{K_{as}}, \{N_b, M, A, B\}_{K_{bs}}$

message3 $S \rightarrow B : M, \{N_a, K_{ab}\}_{K_{as}}, \{N_b, K_{ab}\}_{K_{bs}}$

message4 $B \rightarrow A : M, \{N_a, K_{ab}\}_{K_{as}}$

Principle 7 (可预测值使用原则)



- The use of a predictable quantity (such as the value of a counter) can serve in **guaranteeing newness**.
- But if a predictable quantity is to be effective, **it should be protected** so that an intruder cannot simulate a challenge and later replay the response.

Principle 8 (时戳使用原则)



- If a timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be less than the allowable age of a message deemed to be valid.
- Furthermore, the time maintenance mechanism everywhere becomes part of the trusted computing base.

Principle 9 (密钥新鲜性原则)



- A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised.
- Recent use does not make the key look any better than it would otherwise.

NSSK Protocol



1. $A \rightarrow B : A, B, N_a$
2. $S \rightarrow A : \{ N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}} \}_{K_{as}}$
3. $A \rightarrow B : \{K_{ab}, A\}_{K_{bs}}$
4. $B \rightarrow A : \{ N_b \}_{K_{ab}}$
5. $A \rightarrow B : \{ N_b + 1 \}_{K_{ab}}$

Forcing Reuse of an Old Key

Principle 10 (消息识别和编码准则)



- If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used.
- In the common case where the encoding is protocol dependent, it should be deduce that the message belongs to this protocol, and in fact to a particular run of the protocol.

Principle 11 (信任准则)

- The protocol designers should know which trust relations their protocols depends on, and why the dependence is necessary.
- The reasons for particular trust relations being acceptable should be explicit.



Example for Principle 11



- The Kerberos protocol fails complete if the timestamp on the key server is incurrent.
- Your web-browser comes with a number of public keys for verifying the identity of websites. If these keys are compromised, then you can be tricked by spoof websites.

What we learned::



- Last Lecture: Goals for security protocol
 - To know if a protocol is secure you must know what it aims to achieve.
 - Example: Diffie-Hellman & STS Protocol.
- This Lecture: Attacks and Principles
 - Common types of attacks on protocols.
 - Good design principles for protocol.

Next Time

- Basic security protocol

