Modelling and Analysing Security Protocol: Lecture 4

Attacks and Principles

Tom Chothia
CWI

Today:

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

Some Common Types of Attack

- Eavesdropping
- Modification
- Replay / Preplay
- Man-in-the-Middle
- Reflection
- Denial of Service
- Typing 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.
- Protocols often define against Modification attacks by using encryption for binding.

Replay / Preplay

- The attacker sends a message that it has observed as part of the protocol run.
- Protocols defend against replay attacks by make the message clear so that it cannot be replayed out of context.

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.

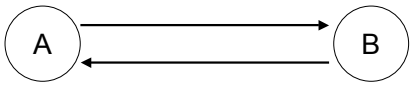
1. $A \rightarrow B : \{ N_a \}_K$
2. $B \rightarrow A : N_a , \{ N_b \}_K$
3. $A \rightarrow B : N_b$

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’. $E(B) \rightarrow A : N_{a2}$

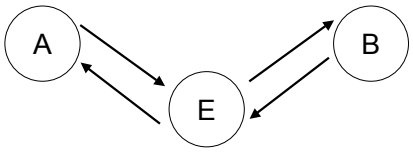
Man-in-the-Middle

- 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.



Denial of Service (DoS)

- Every communication request uses an amount of memory and CPU.
- A DoS attack tries to use up all of a severs 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. $S \rightarrow A : E_{Ka} (N_a , N_s, K_{as})$
3. $A \rightarrow S : \{ N_s \}_{Kas}$

S is particularly vulnerable to a DoS attack because for each connection is 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. $A \rightarrow S : E_{ks}(A, S, Sign_A(N_a,K_{as}))$
2. $S \rightarrow A : \{ N_a \}_{Kas}$

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. $A \rightarrow S$: SYN
2. $S \rightarrow A$: ACK,SYN (add A to the table of connections)
3. $A \rightarrow S$: 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.

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

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

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

Typing Attack Example

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

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

Some Common Types of Attack

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

Good Protocol Design

- The best way to avoid protocol faults is to design them right in the first place.

Principle 0

- The protocol must be efficient
  - No unnecessary encryption
  - Don't include message you don't need.
- 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

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

It should be possible to write down a straight forward 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. $B \rightarrow A : E_A(N_a, N_b , B)$
3. $A \rightarrow B : E_B(N_b)$

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

Meaning of Messages

For instance the Needham-Schroeder Protocol:

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

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

- 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.

Principle 3

- 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. $A \rightarrow B : \{ N_b \}_{Kas}$
4. $B \rightarrow S : \{ A, \{ N_b \}_{Kas} \}_{Kbs}$
5. $S \rightarrow B : \{ N_b \}_{Kbs}$

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. $E(A) \rightarrow B : \{ N_{ba} \}_{Kes}$
   3'. $E \rightarrow B : \{ \boldsymbol{N_{ba}} \}_{Kes}$
4. $B \rightarrow S : \{ A, \{ N_{ba} \}_{Kes} \}_{Kbs}$
   4'. $B \rightarrow S : \{ E, \{ N_{ba} \}_{Kes} \}_{Kbs}$
5. $S \rightarrow B : Fail$
   5'. $S \rightarrow B : \{ N_{ba} \}_{Kbs}$

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. $E(A) \rightarrow B : \{ N_{ba} \}_{Kes}$
   3'. $E \rightarrow B : \{ \boldsymbol{N_{ba}} \}_{Kes}$
4. $B \rightarrow S : \{ A, \{ N_{ba} \}_{Kes} \}_{Kbs}$
   4'. $B \rightarrow S : \{ E, \{ N_{ba} \}_{Kes} \}_{Kbs}$
5. $S \rightarrow B : \{ N_{ba} \}_{Kbs}$

Principle 4

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

Principle 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.
- 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.

$A \rightarrow B : A, Sign_A(T_a, N_a, B, X_a, E_B(Y_a))$

- 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

- 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.

Principle 7

- The use of a predictable quantity (such as the value of a counter) can serve in guaranteeing newness, through a challenge-response exchange.
- 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

- 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

- 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.

Needham-Schroeder Key Establishment Protocol

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

Forcing Reuse of an Old Key

- I spend 1 year breaking a single key (K_{ab}) on a super computer and then trick everyone into using that key.

3. $E \rightarrow B : \{K_{ab}, A\}_{Kbs}$
4. $B \rightarrow E : \{ N_b \}_{Kab}$
5. $E \rightarrow B : \{ N_b + 1 \}_{Kab}$

Principle 10

- 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

- 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.

Today:

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

Homework!

- There is homework that will count 1/6 of your total grade.
- Written exercises, find a couple of protocols errors, correct a protocol, and design a protocol of your own.
- It is due on 28th in class. You may e-mail me questions related to the homework.

Next Time

- No lecture next week (I am presenting an attack on a protocol a conference).
- On the 28th, BAN logic:
  - An framework and software tool for checking protocols.