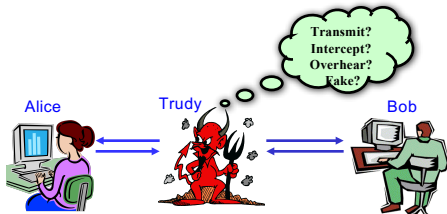


CS458/CS558 Introduction to Computer Security

Chapter 13.2 Authentication Protocols

Authentication Protocols

- Used to convince parties of each other's **identity** and to exchange session keys



Authentication Protocols

- Published protocols are often found to have flaws and need to be modified
- Key issues are
 - ❖ **Confidentiality**
 - To prevent **masquerade** and to prevent **compromise of session keys**, essential identification and session key information must be communicated in encrypted form.
 - ❖ **Timeliness** - to prevent **replay attacks**

Replay Attacks

- A valid signed message is copied and later resent
 - ❖ **Simple replay**: the opponent simply copies a message and replays it later
 - ❖ **Repetition that can be logged**: an opponent can replay a timestamped message within the valid time window
 - ❖ **Repetition that cannot be detected**: may arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
 - ❖ **Backward replay without modification**: a replay back to the sender
 - When using symmetric encryption, the sender cannot easily recognize the difference between messages sent and messages received.

Replay Attacks

- Countermeasures**
 - ❖ Attach a **sequence number** to each message used in an authentication exchange
 - Generally impractical - requires each party to keep track of the last sequence number for each claimant it has dealt with
 - ❖ **Timestamps**: party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time.
 - Needs synchronized clocks

Replay Attacks (Cont.)

Countermeasures

- ❖ **Nonce**: a random number that illustrates the freshness of a session.
 - Party A sends B a nonce and requires that the subsequent response received from B contains the correct nonce value.

Using Symmetric Encryption

- As discussed previously can use a **two-level** hierarchy of keys
- Usually with a trusted **Key Distribution Center (KDC)**
 - ❖ Each party shares own **master key** with KDC
 - ❖ KDC generates **session keys** used for connections between parties
 - ❖ Master keys used to distribute these to them

Needham-Schroeder Symmetric Key Protocol

- Original third-party key distribution protocol
- For session between A B mediated by **KDC**
- Protocol:
 1. **A** → **KDC**: $ID_A || ID_B || N_1$
 2. **KDC** → **A**: $E_{K_a}[K_s || ID_B || N_1 || E_{K_b}[K_s || ID_A]]$
 3. **A** → **B**: $E_{K_b}[K_s || ID_A]$
 4. **B** → **A**: $E_{K_s}[N_2]$
 5. **A** → **B**: $E_{K_s}[f(N_2)]$

Attack: Needham-Schroeder Protocol

3. **A** → **B**: $E_{K_b}[K_s || ID_A]$
 4. **B** → **A**: $E_{K_s}[N_2]$
 5. **A** → **B**: $E_{K_s}[f(N_2)]$
- Suppose that an attacker **X** has been able to compromise an old session key.

Attack: Needham-Schroeder Protocol

3. **A** → **B**: $E_{K_b}[K_s || ID_A]$
 4. **B** → **A**: $E_{K_s}[N_2]$
 5. **A** → **B**: $E_{K_s}[f(N_2)]$
- Suppose that an attacker **X** has been able to compromise an old session key.
 - **X** can impersonate **A** and trick **B** into using the old key by simply replaying step 3.
 - Unless **B** remembers indefinitely all previous session keys used with **A**, **B** will be unable to determine that this is a replay.
 - **X** then **intercepts the step 4** and sends bogus messages to **B** that appear to **B** to come from **A** using an authenticated session key

Solution: Needham-Schroeder Protocol

- Use a **timestamp T** that assures **A** and **B** that the session key has only just been generated.
- Revised protocol:
 1. **A** → **KDC**: $ID_A || ID_B || N_1$
 2. **KDC** → **A**: $E_{K_a}[K_s || ID_B || N_1 || E_{K_b}[K_s || ID_A || T]]$
 3. **A** → **B**: $E_{K_b}[K_s || ID_A || T]$
 4. **B** → **A**: $E_{K_s}[N_2]$
 5. **A** → **B**: $E_{K_s}[f(N_2)]$

Timestamp

- Principals can verify the timeliness by checking:

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$
 - Δt_1 : The estimated normal discrepancy between the KDC's clock and the local clock (principals' clock)
 - Δt_2 : The expected network delay time
- Need to **synchronize clock**

Suppress-Replay Attacks

- Suppress-replay attacks**: when the sender's clock is ahead of the receiver's clock, the opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the receiver's site.
 - Replaying a message to purchase oil stocks one day later could result in a purchase when the stock price had already changed significantly.

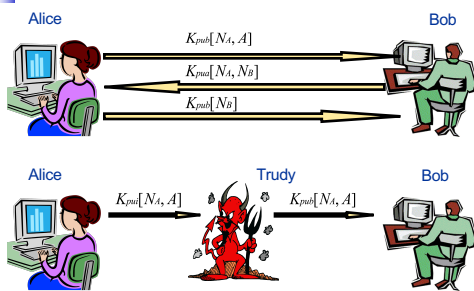
Suppress-Replay Attacks

- Suppress-replay attacks**: when the sender's clock is ahead of the receiver's clock, the opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the receiver's site.
 - Replaying a message to purchase oil stocks one day later could result in a purchase when the stock price had already changed significantly.
- Countermeasure**:
 - Enforce the requirement that parties regularly check their clocks against the KDC's clock.
 - Rely on handshaking protocols using **nonces**.

Using Public-Key Encryption

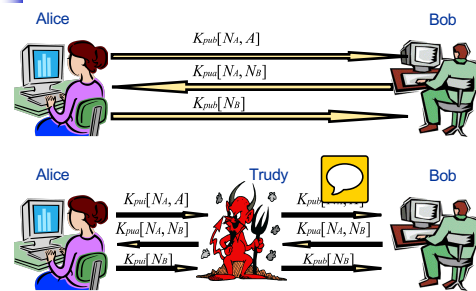
- Have a range of approaches based on the use of public-key encryption
- Need to ensure have correct public keys for other parties
- Various protocols exist using timestamps or nonces

Needham-Schroeder Public Key Protocol

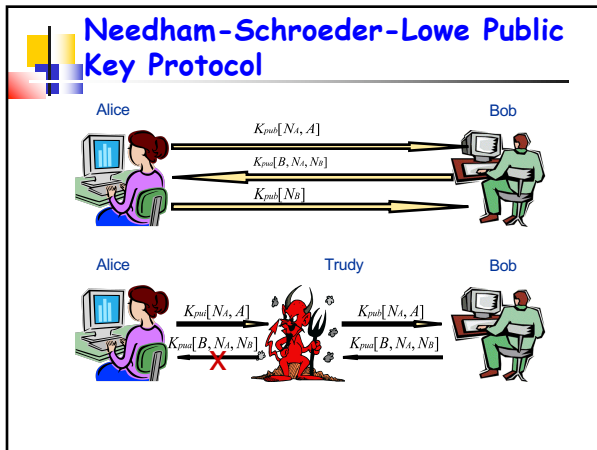


[Lowe, 1995]

Needham-Schroeder Public Key Protocol



[Lowe, 1995]




- ### One-Way Authentication
- Required when sender and receiver are **not** in communications at same time (eg. **email**)
 - Have header in clear so can be delivered by email system
 - May want contents of body protected & sender authenticated

- ### Using Symmetric Encryption
- Can refine use of KDC but cannot have exchange of nonces:
1. A → KDC: $ID_A \parallel ID_B \parallel N_I$
 2. KDC → A: $E_{K_a}[K_s \parallel ID_B \parallel N_I \parallel E_{K_b}[K_s \parallel ID_A]]$
 3. A → B: $E_{K_b}[K_s \parallel ID_A] \parallel E_{K_s}[M]$

- ### Using Symmetric Encryption
- Can refine use of KDC but cannot have exchange of nonces:
1. A → KDC: $ID_A \parallel ID_B \parallel N_I$
 2. KDC → A: $E_{K_a}[K_s \parallel ID_B \parallel N_I \parallel E_{K_b}[K_s \parallel ID_A]]$
 3. A → B: $E_{K_b}[K_s \parallel ID_A] \parallel E_{K_s}[M]$
- Guarantees that only the **intended recipient** of a message will be able to read it.
 - Does not protect against **replays**
 - Could rely on timestamp in message, though email delays make this problematic

- ### Assignment 4
- Due: **April 16th (Monday)**
 - Done individually or by a group of two students
 - Two choices: rootkit or PGP (choose **ONE** of them)

- ### Assignment 4: Choice I
- choice I: Rootkit**
 - Find and download a windows rootkit that enables the attacker to hide files or processes, and demonstrate how to do it.
 - You will need to install a **virtual machine** (e.g. virtualbox) and execute the rootkit inside the virtual machine.
 - Demo: **April 16th at 625** (or during Nikhil's office hours before April 16th)



Assignment 4: Choice II

- ◆ choice II: PGP
- ◆ Find **PGP software** that supports confidentiality and digital signature, and show how to use PGP to provide confidentiality and digital signature.
- ◆ You can choose **any email client** and **any PGP software**.
- ◆ Demo: **April 16th at 625** (or during Nikhil's office hours before April 16th)