Lecture 11

Protocols (Continued)

Chapters 9 and 11 in KPS
Key Distribution Center (KDC) or Trusted Third Party (TTP)

- Alice and Bob communicate using $K$ as a short-term (session) key for encryption and/or data integrity.

Note:
- $\text{Msg2}$ is not tied to $\text{Msg1}$
- $\text{Msg1}$ is possibly old
- $\text{Msg2}$ is possibly old and so is $\text{Msg3}$
- Bob and Alice don’t authenticate each other!

\[ \text{KDC generates fresh } K \]

\[ K(\text{X}) = \text{Encryption of } \text{X with key } K \]

\[ \text{Msg1: } K_A(\text{A,B}) \]

\[ \text{Msg2: } K_A(K, K_B(A, K)) \]

\[ \text{Msg3: } K_B(A, K) \]

- Bob obtains $K$ and knows to use as a key for communicating with Alice.
A Typical Key Distribution Scenario

\[ E_K[X] = \text{Encryption of } X \text{ with key } K \]

1. Request \( |B|N_1 \)
2. \( E_{K_a}[K_s | \text{Request} | N_1 | E_{K_b}(K_s, A)] \)
3. \( E_{K_b}[K_s, A] \)
4. \( E_{K_s}[A, N_2] \)
5. \( E_{K_s}[f(N_2)] \)

Notes:
- Msg2 is tied to Msg1
- Msg2 is fresh/new
- Msg3 is possibly old *
- Msg1 is possibly old (KDC doesn’t authenticate Alice)
- Bob authenticates Alice
- Bob authenticates KDC
- Alice DOES NOT authenticate Bob
Public Key Distribution

• General schemes:
  • Public announcement (e.g., in a newsgroup or email message)
    • Can be forged
  • Publicly available directory
    • Can be tampered with
  • Public-key certificates (PKCs) issued by trusted off-line Certification Authorities (CAs)
Certification Authorities

- Certification authority (CA): binds public key to a specific entity
- Each entity (user, host, etc.) registers its public key with CA.
  - Bob provides “proof of identity” to CA.
  - CA creates certificate binding Bob to this public key.
  - Certificate containing Bob’s public key digitally signed by CA:
    CA says: “this is Bob’s public key”
When Alice wants to get Bob’s public key:
- Get Bob’s certificate (from Bob or elsewhere)
- Using CA’s public key verify the signature on Bob’s certificate
- Check for expiration
- Check for revocation (we’ll talk about this later)
- Extract Bob’s public key
A Certificate Contains

• Serial number (unique to issuer)
• Info about certificate owner, including algorithm and key value itself (not shown)

• info about certificate issuer
• valid dates
• digital signature by issuer
A Sample Certificate (1/2)
A Sample Certificate (2/2)
Back to Protocols

\[\{X\}_K = \text{Encryption of } X \text{ with key } K\]

1. A $\rightarrow$ T: A, B, $N_A$
2. T $\rightarrow$ A: $\{N_A, B, K, \{K, A\}_K\}_K$
3. A $\rightarrow$ B: $\{K, A\}_K$
4. B $\rightarrow$ A: $\{N_B\}_K$
5. A $\rightarrow$ B: $\{N_B-1\}_K$
Denning-Sacco Attack: suppose Eve recorded an old session for which session key $K'$ is known to her:

1. $A \rightarrow T$: $A, B, N_A$
2. $T \rightarrow A$: $\{N_A, B, K', \{K', A\}_{K_B}\}_{K_A}$
3. $A \rightarrow B$: $\{K', A\}_{K_B}$

At a later time:

1. $E \rightarrow B$: $\{K', A\}_{K_B}$
2. $B \rightarrow E$: $\{N_B\}_{K'}$
3. $E \rightarrow B$: $\{N_{B-1}\}_{K'}$
Fixing the Attack

• Bob has no guarantees about the freshness of the message in step 3.
• Eve exploits this to impersonate Alice to Bob - old session keys are useful.
• Can be fixed by adding timestamps:
  • Limits usefulness of old session keys
  • Eve’s attack becomes:

3: E → B: $\{K', T', A\}_{KB}$

attack is now thwarted because $T'$ is stale
**PK-based Needham-S Schroeder Protocol**

1. \{A, B\}
2. \{PK_b, B\}_{SK_T}
3. \[N_a, A\]_{PK_b}
4. \{B, A\}_{SK_T}
5. \{PK_a, A\}_{SK_T}
6. \[N_a, N_b\]_{PK_a}
7. \[N_b\]_{PK_b}

- \text{CERT}_B = \text{Message 2, CERT}_A = \text{Message 5}
- \text{PK}_A: Alice's public key, \text{PK}_B: Bob's public key
- \text{SK}_T: TTP's secret (private) key used for signing
- Everyone knows TTP's public key \text{PK}_T

\([X]_K = \text{Encryption of } X \text{ with key } K\)
Another Attack

• 1, 2, 4, 5: Delivery of public key
• Does not guarantee freshness of the public key

• How to solve it?
  • Timestamp in messages 2 and 5 or challenges in messages 1&2 and 4&5
  • Public Key Certificate: assign expiration time/data to each certificate (messages 2 and 5)
PK-based Denning-Sacco Attack

1. A, B
2. Cert_A, Cert_B
3. Cert_A, Cert_B, [ {K_{AB}, T_A}_{SK_A} ]_{PK_B}
4. Secure communication with K_{AB}

3’. Cert_A, Cert_C, [ {K_{AB}, T_A}_{SK_A} ]_{PK_C}
4’. Secure communication with K_{AB}

Cert_A = {PK_A, A}_{SK_T}
Cert_B = {PK_B, B}_{SK_T}
Cert_C = {PK_C, C}_{SK_T}

Thinks she is talking to A
Pretends to be A
Lowe’s Attack
(Impersonation by Interleaving)

Original

3. $A \rightarrow B$: $[N_a, A]_{PKb}$
6. $B \rightarrow A$: $[N_a, N_b]_{PKa}$
7. $A \rightarrow B$: $[N_b]_{PKb}$

Fix

3. $A \rightarrow B$: $[N_a, A]_{PKb}$
6. $B \rightarrow A$: $[B, N_a, N_b]_{PKa}$
7. $A \rightarrow B$: $[N_b]_{PKb}$

Attack $E$ ‘plays’ $A$:

1.3. $A \rightarrow E$: $[N_a, A]_{PKe}$
2.3. $E \rightarrow B$: $[N_a, A]_{PKb}$
2.6. $B \rightarrow E$: $[N_a, N_b]_{PKa}$
1.6. $E \rightarrow A$: $[N_a, N_b]_{PKa}$
1.7. $A \rightarrow E$: $[N_b]_{PKe}$
2.7. $E \rightarrow B$: $[N_b]_{PKb}$
Fixed PK-based Needham-Schroeder Protocol

1. \(\{A, B\}\)

2. \(\{PK_b, B\}_{sk}\)

3. \([N_a, A]_{PK_b}\)

4. \(\{B, A\}_{sk}\)

5. \(\{PK_a, A\}_{sk}\)

6. \([B, N_a, N_b]_{PK_a}\)

7. \([N_b]_{PK_b}\)
Reflection Attack and a Fix

• **Original Protocol**
  1. $A \rightarrow B : \ r_A$
  2. $B \rightarrow A : \ \{ r_A, r_B \}_K$
  3. $A \rightarrow B : \ r_B$

• **Attack**
  1. $A \rightarrow E : \ r_A$
  2. $E \rightarrow A : \ r_A : \text{Starting a new session}$
  3. $A \rightarrow E : \ \{ r_A, r_A' \}_K : \text{Reply to (2)}$
  4. $E \rightarrow A : \ \{ r_A, r_A' \}_K : \text{Reply to (1)}$
  5. $A \rightarrow E : \ r_A'$

Solutions?
• Use 2 different uni-directional keys $k''$ ($A \rightarrow B$) and $k'$ ($B \rightarrow A$)
• Remove symmetry (direction, msg identifiers)
Interleaving Attacks

- Protocol for Mutual Authentication

1. A → B : A, r_A,
2. B → A : r_B, \{ r_B, r_A, A \}_{SK_B}
3. A → B : r_A', \{ r_A', r_B, B \}_{SK_A}

- Attack

1. E → B : A, r_A
2. B → E : r_B, \{ r_B, r_A, A \}_{SK_B}
3. E → A : B, r_B
4. A → E : r_A', \{ r_A', r_B, B \}_{SK_A}
5. E → B : r_A', \{ r_A', r_B, B \}_{SK_A}

- Attack due to symmetric messages (2), (3)
Lessons learned?

- Designing **secure** protocols is hard. There are **many** documented failures in the literature.
- Good protocols are already standardized (e.g., ISO 9798, X.509, ...) – use them!
- The problem of verifying security gets much harder as protocols get more complex (more parties, messages, rounds).
If interested in knowing more, read the paper:

“Programming Satan’s Computer”
by Anderson and Needham

available at:

http://www.cl.cam.ac.uk/~rja14/Papers/satan.pdf
Some More Examples of Secure Protocol Examples
Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)

Choose random $v$

$K_{ab} = (y_b)^v \mod p$

$SIG_{alice} = \{y_a, y_b\}^{alice}$

$y_a = a^v \mod p$

$CERT_{bob}, y_b, SIG_{bob}$

Choose random $w$, Compute

$K_{ba} = (y_a)^w \mod p$

$y_b = a^w \mod p$

$SIG_{bob} = \{y_b, y_a\}^{Bob}$

$CERT_{alice}, SIG_{alice}$
x.509 Authentication & Key Distribution Protocols

One-way

$$\{1, t_a, r_a, B, other_a, [K_{ab}]_{PK_B} \}^{SK_A}_{SK_A}$$

Two-way

$$\{2, t_a, r_a, B, other_a, [K_{ab}]_{PK_B} \}^{SK_A}_{SK_A}$$

$$\{2, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A} \}^{SK_B}_{SK_B}$$

Three-way

$$\{3, t_a, r_a, B, other_a, [K_{ab}]_{PK_B} \}^{SK_A}_{SK_A}$$

$$\{3, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A} \}^{SK_B}_{SK_B}$$

$$\{3, r_b \}^{SK_A}_{SK_A}$$