Lecture 14

Protocols...

Key Distribution Center (KDC) or Trusted Third Party (TTP)

Alice obtains R1

Bob obtains R1 and knows to use as a key for communicating with Alice

KDC generates R1

Msg1: $K_A(B)$

Msg2: $K_A(R1, K_B(A, R1))$

Msg3: $K_B(A, R1)$

- Alice and Bob communicate using R1 as a short-term (session) key for encryption and/or data integrity
- Note:
  - Msg2 is not tied to Msg1
  - Msg1 is possibly old
  - Msg2 is possibly old and so is Msg3
  - Bob and Alice don’t authenticate each other!
**A Typical Key Distribution Scenario**

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

**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 (recall discussion in class)

---

**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:
    
    \[ \text{CA says: "this is Bob's public key"} \]

Bob's public key

Bob's identifying information

digital signature

PK_B

CA private key

SK_CA

Certificate for Bob's public key, signed by CA

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

Back to protocols
**Needham-Schroeder Protocol (1978):**

1st distributed security protocol

1. \( A \rightarrow T: \) \( A, B, N_A \)
2. \( T \rightarrow A: \) \( \{N_A, B, K, \{K, A\}_K_B\}_K_A \)
3. \( A \rightarrow B: \) \( \{K, A\}_K_B \)
4. \( B \rightarrow A: \) \( \{N_B\}_K \)
5. \( A \rightarrow B: \) \( \{N_B-1\}_K \)

**Security?**

**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:

  $3: E \to B: \{K', T', A\}_{K_B}$

  attack is now thwarted because $T'$ is stale

PK-based Needham-Schroeder protocol

- $\text{CERT}_B = \text{Message 2}$, $\text{CERT}_A = \text{Message 5}$
- $\text{PK}_A$: 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 $\text{PK}_T$
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} \cdot T_A)_SK_A ]_{PK_B}
4. Secure communication with K_{AB}

C
3'. Cert_A, Cert_C, [ (K_{AB} \cdot T_A)_SK_A ]_{PK_C}
4'. Secure communication with K_{AB}
Lowe’s Attack
(Impersonation by interleaving)

Original
3. \( A \rightarrow B: [N_a, A]_{PK_b} \)
6. \( B \rightarrow A: [N_a, N_b]_{PK_a} \)
7. \( A \rightarrow B: [N_b]_{PK_b} \)

Fix
3. \( A \rightarrow B: [N_a, A]_{PK_b} \)
6. \( B \rightarrow A: [B, N_a, N_b]_{PK_a} \)
7. \( A \rightarrow B: [N_b]_{PK_b} \)

Attack \( E \) ‘plays’ \( A \):
1.3. \( A \rightarrow E: [N_a, A]_{PK_a} \)
2.3. \( E \rightarrow B: [N_a, A]_{PK_b} \)
2.6. \( B \rightarrow E: [N_a, N_b]_{PK_a} \)
1.6. \( E \rightarrow A: [N_a, N_b]_{PK_a} \)
1.7. \( A \rightarrow E: [N_b]_{PK_a} \)
2.7. \( E \rightarrow B: [N_b]_{PK_b} \)

PK-based Needham-Schroeder protocol

1. \((A, B)\)
2. \((PK_b, B)_{SK} \)
3. \([N_a, A]_{PK_b} \)
4. \((B, A)\)
5. \((PK_a, A)_{SK} \)
6. \([N_a, N_b]_{PK_a} \)
7. \([N_b]_{PK_b} \)
Reflection Attack and a fix

- **Original protocol**
  1. A → B : \( r_A \)
  2. B → A : \( \{ r_A, r_B \}_K \)
  3. A → B : \( r_B \)

- **Attack**
  1. A → E : \( r_A \)
  2. E → A : \( r_A : \) Starting a new session
  3. A → E : \( \{ r_A, r'_A \}_K : \) Reply to (2)
  4. E → A : \( \{ r_A, r'_A \}_K : \) Reply to (1)
  5. A → 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 Secure Protocol examples

Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)

Choose random $v$

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

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

$y_j = a^v \mod p$

$CERT_{alice}, SIG_{alice}$

Choose random $w$, Compute

$K_{wb} = (y_w)^w \mod p$

$y_y = a^w \mod p$

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

$SIG_{bob} = \{ y_b, y_a \}_{bob}$
x.509 Authentication & Key Distribution Protocols

One-way

\[ \{1, t_a, r_a, B, other_a, [K_{ab}]_{PK_a}\}^{SK_a} \]

Two-way

\[ \{2, t_a, r_a, B, other_a, [K_{ab}]_{PK_a}\}^{SK_a} \]
\[ \{2, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_b}\}^{SK_b} \]

Tree-way

\[ \{3, t_a, r_a, B, other_a, [K_{ab}]_{PK_a}\}^{SK_a} \]
\[ \{3, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_b}\}^{SK_b} \]
\[ \{3, r_b\}^{SK_b} \]