Authentication Protocols

It's Not (only) What You Say,

But How You Say It.
Cryptographic authentication, revisited

Alice, knock-knock

Alice

Bob

$X = F(K, R)$

$K = \text{shared key between Alice and Bob}$
Modified cryptographic authentication

Alice

Alice, knock-knock

K_{Alice-Bob}\{R\}

R

Bob
One-pass authentication

Alice, $K_{Alice-Bob}\{Bob, Timestamp T\}$

Alice, timestamp $T$, $F(K_{Alice-Bob}, T)$
Mediated Authentication

\[ T_1 = K_{Alice} \{ \text{Use } K_{AB} \text{ for Bob} \} \]

\[ T_2 = K_{Bob} \{ \text{Use } K_{AB} \text{ for Alice} \} \]

Alice, \( T_2 \), “proof(Alice)”
Needham-Schroeder

Nonce $N_1$, Alice for Bob

$K_{Alice} \{N_1, \text{“Bob”}, K_{AB}, T\}$

$T = K_{Bob} \{K_{AB}, \text{“Alice”}\}$

$T, K_{AB} \{N_2\}$

$T, K_{AB} \{N_2-1, N_3\}$

$K_{AB} \{N_3-1\}$
Otway-Rees

$N_C, A, B, K_A\{N_A, N_C, A, B\}$

$K_A\{N_A, N_C, A, B\}$

$K_B\{N_B, N_C, A, B\}$

$N_C, K_A\{N_A, K_{AB}\}, K_B\{N_B, K_{AB}\}$

$K_A\{N_A, K_{AB}\}$

$K_{AB}\{\text{anything recognizable}\}$
Kerberos protocol

N₁, Alice for Bob

Kₐ₁{N₁, B, Kₐₐ, T}

T, Kₐₐ{time}

Kₐₐ{time + 1}

Where T = Kₐ₉{Kₐₐ, A, “timestamp”}
Protocol Design Principles

“A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools. -- Douglas Adams”
Rules of good protocol practice

• Make clear the meaning of each protocol message
  – In particular, make explicit the circumstances under which a message should be considered valid, or invalid.

• Naming: If the name (of a principal) is essential to the interpretation of the message, the name should in the message
  – Integrity protection is required
Example (Naming)

• Woo-Lam
1. $A \rightarrow B$: $A$
2. $B \rightarrow A$: $N_B$
3. $A \rightarrow B$: $K_{AS}\{N_B\}$
4. $B \rightarrow S$: $K_{BS}\{A,K_{AS}\{N_B\}\}$
5. $S \rightarrow B$: $K_{BS}\{N_B\}$

• Attack:
1. $C \rightarrow B$: $A$; $C \rightarrow B$: $C$
2. $B \rightarrow A$: $N_1$; $B \rightarrow C$: $N_2$
3. $C \rightarrow B$: $K_{CS}\{N_1\}$ (twice)
4. $B \rightarrow S$: $K_{BS}\{A,K_{CS}\{N_1\}\}$, $K_{BS}\{C,K_{CS}\{N_1\}\}$
5. $S \rightarrow B$: $K_{BS}\{K_{AS}^{-1}\{K_{CS}\{N_1\}\}\}$, $K_{BS}\{N_1\}$
Woo-Lam simplified

1. $A \rightarrow B$: $A$
2. $B \rightarrow A$: $N_B$
3. $A \rightarrow B$: $K_{AS}\{N_B\}$
4. $B \rightarrow S$: $K_{BS}\{A, K_{AS}\{N_B\}\}$
5. $S \rightarrow B$: $K_{BS}\{A, N_B\}$

1. $A \rightarrow B$: $A$
2. $B \rightarrow A$: $N_B$
3. $A \rightarrow B$: $K_{AS}\{N_B\}$
4. $B \rightarrow S$: $A, B,$ $K_{AS}\{N_B\}$
5. $S \rightarrow B$: $K_{BS}\{A, N_B\}$
Uses of encryption

• Confidentiality
• Authentication
• Binding different parts of a message:
  – $K\{X, Y\}$ is different from $K\{X\}$ and $K\{Y\}$
• To produce random numbers
  – To produce unpredictable inputs that provide liveliness guarantees
• Since encryption has many usages, be clear about which uses is meant
Example: Kerberos (original)

- A → S: A, B
- S → A: $K_{AS}\{T_S, L, K_{AB}, B, K_{BS}\{T_S, L, K_{AB}, A}\}$
- A → B: $K_{BS}\{T_S, L, K_{AB}, A\}, K_{AB}\{A, T_A\}$
- B → A: $K_{AB}\{T_A + 1\}$

- Double encryption of the “ticket”:
  - It accomplishes a “liveliness” guarantee -- A must have seen the ticket after time $T_S$. (Double encryption has been eliminated from Kerberos.) Again in Kerberos, the sequential nature of messages can be deduced from the use of good timestamps. (Kerberos assume synchronized clocks.)
Randomness, nonces and timestamps

• Freshness is important in authentication
  – Prevention of re-play attacks

• Consistency is crucial:
  – Prevention of interleaved impersonation attacks

• These are different things:
  – Temporal succession
  – Association
Example of Otway-Rees

- **A → B**: $N, A, B, E$
  - $E = K_{AS}\{N_A, N, A, B\}$
- **B → S**: $N, A, B, E, F$
  - $F = K_{BS}\{N_B, N, A, B\}$
- **S → B**: $N, G, H$
  - $G = K_{AS}\{N_A, K_{AB}\}$
  - $H = K_{BS}\{N_B, K_{AB}\}$
- **B → A**: $N, G$
- **A → B**: $A, B, N_A$
- **B → S**: $A, B, N_A, N_B$
- **S → B**: $E = K_{AS}\{N_A, A, B, K_{AB}\}$
  - $F = K_{BS}\{N_B, A, B, K_{AB}\}$
- **B → A**: $E$
Summary

- Easy to get things wrong when designing cryptographic protocols:
  - Too many ways to attack a protocol in practice
  - Human tendency to see implicit meaning into things that have to be interpreted by machines (which can only handle explicit information)

- Prescription:
  - Follow best practices (guides such as Abadi et al. help)
  - Make clear what is meant by each message, and how this meaning will be deduced in the receiver’s end.
  - Know that small changes can break a good scheme.