Some attacks on protocols

Needham-Schroeder

- Authentication/key agreement protocol between two parties

- Goal: secure channel between Alice and Bob, and each can trust the other's identity

- There is a public-key protocol, and also a secret-key protocol using a trusted Key Server

- Used as basis for real-world authentication systems (Kerberos is derived from secret-key NS protocol)

Needham-Schroeder I

1. Alice→Bob: $\{N_a\}_{EB}$

2. Bob→Alice: $\{N_a, N_b\}_{EA}$

3. Alice→Bob: $\{N_b\}_{EB}$

- Alice, Bob both have public keys E_B, E_A
- N_a, N_b are nonces

Needham-Schroeder I

1. Alice→Bob: $\{N_a\}_{EB}$

2. Bob→Alice: $\{N_a, N_b\}_{EA}$

3. Alice→Bob: $\{N_b\}_{EB}$

- Bob knows he's talking to Alice (why?) and Alice knows she's talking to Bob (why?)
- Can use hash(N_a, N_b) as a session key

Needham-Schroeder I

1. Alice→Bob: $\{N_a\}_{EB}$

2. Bob→Alice: $\{N_a,N_b\}_{EA}$

3. Alice→Bob: $\{N_b\}_{EB}$

- Eve wants to impersonate Alice to Bob
- Eve is another legitimate principal in the system with her own key Ee

Needham-Schroeder I

1. Alice→Eve: $\{N_a\}_{Ee}$
2. Eve→Alice: $\{N_a,N_b\}_{EA}$
3. Alice→Eve: $\{N_b\}_{Ee}$

1. "Alice"→Bob: $\{N_a\}_{EB}$
2. Bob→Alice: $\{N_a,N_b\}_{EA}$
3. "Alice"→Bob: $\{N_b\}_{EB}$

- Can exploit multiple instances of same protocol!
- This violates the "principle of explicitness"

Needham-Schroeder I

1. Alice→Bob: $\{N_a\}_{EB}$

2. Bob→Alice: $\{B, N_a, N_b\}_{EA}$

3. Alice→Bob: $\{N_b\}_{EB}$

- Datagram 3 can still be conflated for datagram 2, or replayed from other protocol instances
- Best to label everything, e.g. 1. A→B: $\{1, A, B, N_a\}_{EB}$

Needham-Schroeder II

- Symmetric cryptography only
- Every principal has a secret key to communicate with a Key Server (K_{AS} for Alice⇔Server, K_{BS} for Bob⇔Server, etc)
- Alice wants to initiate communication with Bob, asks Server to set up a session key K_{AB}

Needham-Schroeder II

1. Alice→Server: A, B, N_a

2. Server→Alice: $\{N_a, K_{AB}, B, \{A, K_{AB}\}_{KBS}\}_{KAS}$

3. Alice→Bob: $\{A, K_{AB}\}_{KBS}$

4. Bob→Alice: $\{N_b\}_{KAB}$

5. Alice→Bob: $\{N_b - 1\}_{KAB}$

Needham-Schroeder II

1. Alice→Server: A,B,N_a

2. Server→Alice: $\{N_a,K_{AB},B,\{A,K_{AB}\}_{KBS}\}_{KAS}$

3. Alice→Bob: $\{A,K_{AB}\}_{KBS}$

4. Bob→Alice: $\{N_b\}_{KAB}$

5. Alice→Bob: $\{N_b\text{-}1\}_{KAB}$

Why no encryption?

Why the nonce?

Why subtract 1?

Needham-Schroeder II

3. Alice→Bob: $\{A, K_{AB}\}_{KBS}$

4. Bob→Alice: $\{N_b\}_{KAB}$

5. Alice→Bob: $\{N_b\text{-}1\}_{KAB}$

6. Eve→Bob: $\{A, K_{AB}\}_{KBS}$

7. Bob→Alice: $\{N_b\}_{KAB}$

8. "Alice"→Bob: $\{N_b\text{-}1\}_{KAB}$

- Eve can just wait for a session key that gets stale or is compromised, and replay the datagram to Bob

Wide-mouthed frog

1. Alice→Server: $\{B, K_{AB}, Time\}_{KAS}$

2. Server→Bob: $\{A, K_{AB}, Time^*\}_{KBS}$

- Alice makes up her own session key (server's work is minimized)
- Every packet has a fresh timestamp, so Time* is updated from the value of Time

Wide-mouthed frog

1. Alice→Server: $\{B,K_{AB},Time\}_{KAS}$

2. Server→Bob: $\{A,K_{AB},Time^*\}_{KBS}$

3.Eve→Server: $\{A,K_{AB},Time^*\}_{KBS}$

4.Server→Alice: $\{B,K_{AB},Time^{**}\}_{KAS}$

- Eve reflects packet back to server, who updates the timestamp

- Can keep key packet fresh indefinitely!

How can we know a protocol is secure?

- Formal methods (mathematical proof) to establish that security properties are guaranteed at each step
- Limits to formal methods: they require assumptions about protocols that may not be true in reality
- Security guarantees may not capture an attack beyond our assumptions of the attacker (e.g. denial of service attacks)

Example: Otway-Rees

1. Alice→Bob: M,A,B, $\{N_a,M,A,B\}_{KAS}$

2. Bob→Server: M,A,B, $\{N_a,M,A,B\}_{KAS}$, $\{N_b,M,A,B\}_{KBS}$

3. Server→Bob: M,$\{N_a,K_{AB}\}_{KAS}$, $\{N_b,K_{AB}\}_{KBS}$

4. Bob→Alice: M,$\{N_a,K_{AB}\}_{KAS}$

M is a session ID

- In step 2, the server verifies that both packets have the same session ID, and same principals.

Example: Otway-Rees

1. Alice→Bob: M,A,B, $\{N_a,M,A,B\}_{KAS}$

4. Eve→Alice: M,$\{N_a,M,A,B\}_{KAS}$

reflection attack!

- Alice expects $\{N_a,K_{AB}\}_{KAS}$, we assume this won't work because the datagrams don't match
- But what if these parcels are delivered as packets?

N_a	M	A	B

N_a	K_{AB}

How can we know a protocol is secure?

- Formal methods (mathematical proof) to establish that security properties are guaranteed at each step

- Limits to formal methods: they require assumptions about protocols that may not be true in reality

BAN logic

- Burrows, Abadi and Needham
- States basic premises about protocol (shared secret keys, etc)
- Uses simple axioms, or production rules, to derive security facts from those premises
- Section 3.8 of textbook (2nd ed.)

BAN logic notation

- $A \mid\equiv X$ Alice believes statement X
- $A \triangleleft M$ Alice sees message M
- $A \mid\Rightarrow X$ Alice is an authority on X
- $A \mid\sim M$ Alice has (once) stated message M
- $\#M$ Message M is fresh (recent)
- $A \Leftrightarrow_K B$ Alice and Bob share secret key K

BAN logic rules

- The "message meaning" rule:

  if $A \mid\equiv (A \Leftrightarrow_K B)$ and $A \triangleleft \{M\}_K$ then $A \mid\equiv (B \mid\sim M)$

- The "nonce verification" rule:

  if $A \mid\equiv (B \mid\sim M)$ and $\#M$, then $A \mid\equiv (B \mid\equiv M)$

- The "jurisdiction" rule:

  if $A \mid\equiv (B \mid\equiv M)$ and $A \mid\equiv (B \mid\Rightarrow M)$ then $A \mid\equiv M$

BAN logic rules

- The "message meaning" rule:

  if $A \mid\equiv (A \overset{K}{\Leftrightarrow} B)$ and $A \triangleleft \{M\}_K$ then $A \mid\equiv (B \mid\sim M)$
  "Alice believes $\{M\}_K$ is originally from Bob"

- The "nonce verification" rule:

  if $A \mid\equiv (B \mid\sim M)$ and $\#M$, then $A \mid\equiv (B \mid\equiv M)$
  "Alice believes Bob means M, if he says M and it's fresh"

- The "jurisdiction" rule:

  if $A \mid\equiv (B \mid\equiv M)$ and $A \mid\equiv (B \mid\Rightarrow M)$ then $A \mid\equiv M$

  "If Bob believes M and Bob is an expert, Alice believes it too."

Example: Wide Mouthed Frog

1. Alice$\rightarrow$Server: $\{B, K_{AB}, Time\}_{KAS}$

2. Server$\rightarrow$Bob: $\{A, K_{AB}, Time^*\}_{KBS}$

- We start with facts:
  $A \Leftrightarrow_{KAS} S$ $\quad$ $B \Leftrightarrow_{KBS} S$
  $S \mid\Rightarrow (A \Leftrightarrow_K B)$ (trust server WRT keys)
  $A \mid\Rightarrow (A \Leftrightarrow_K B)$ (Alice makes her own key for Bob)

Example: Wide Mouthed Frog

1. Alice→Server: $\{B, K_{AB}, Time\}_{KAS}$
2. Server→Bob: $\{A, K_{AB}, Time^*\}_{KBS}$

1. $S \triangleleft \{B, K, Time\}_{KAS}$
2. $S \mid\equiv (A \mid\sim (A \Leftrightarrow_K B))$ (message meaning rule)
3. $S \mid\equiv (A \mid\equiv (A \Leftrightarrow_K B))$ (because message is fresh)
4. $S \mid\equiv (A \Leftrightarrow_K B)$ (because Alice is authority on key)

Example: Wide Mouthed Frog

1. Alice→Server: $\{B, K_{AB}, Time\}_{KAS}$

2. Server→Bob: $\{A, K_{AB}, Time^*\}_{KBS}$

1. $B \triangleleft \{A, K, Time\}_{KBS}$

2. $B \mid\equiv (S \mid\sim (A \Leftrightarrow_K B))$ (message meaning rule)

3. $B \mid\equiv (S \mid\equiv (A \Leftrightarrow_K B))$ (because message is fresh)

4. $B \mid\equiv (A \Leftrightarrow_K B)$ (because Server is authority on keys)