Aalto University

Network Security:
Replay and freshness

Tuomas Aura, Aalto University

CS-E4300 Network security

Outline

1. Alice and Bob
2. Replay and freshness
3. Timestamp
4. Sequence number
5. Nonce

These basic concepts were also covered in the introductory Information Security course

The first broken protocol

- Please meet Alice and Bob!
- Alice sends a signed message to Bob:

  A → B: M, S_A(M)   // Example: S_A("Attack now!")

  Assumption: Alice and Bob know each others' public keys

- What things are wrong with this protocol?

If you want to learn, stop here and think a few minutes before looking at the solution

Being explicit

- Should include recipient id:
  A → B: B, M, S_A(B,M)   // Example: S_A("Bob, attack now!")
- Include important information, such as endpoint identities, explicitly in the authenticated message

- What about Alice's identity?
- What else is wrong with this protocol?

Replay and freshness

$A \rightarrow B$: B, M, S_A(B, M) // S_A("Bob, attack now!")

- Replay attack: attacker sniffs the original message and sends it again on the next day
- Authentication is usually not enough in network security! Need to also check freshness of the message
  - Fresh = sent recently, not received before (exact definition depends on the application)
  - Freshness mechanisms: timestamp, nonce, sequence number

Timestamps

- Checking freshness with A's timestamp:

  $A \rightarrow B$: $B, T_A, M, S_A(B, T_A, M)$

  Example: S_A("2019-10-28 14:15 GMT", "Bob, attack now!")

- Timestamp implementations:
  - Sender's clock value, UTC
  - Expiration time
  - Validity start and end times

Timestamp limitations

- Timestamp requires clocks at the sender and receiver
- Timestamp requires secure clock synchronization
  - Secure fine-grained synchronization is difficult to implement
  - Loose synchronization (minutes or over 24 h) is easier
- Clock must never turn back
- Problematic in IoT devices, smartcards, locks etc.

> When can timestamps be used without clock synchronization?

- Fast replays while the timestamp is fresh:

$S_A(B, T_A,$ "Transfer £10."$)$, $S_A(B, T_A,$ "Transfer £10."$)$

  - Solutions: idempotent operations, duplicate detection with sequence numbers

Sequence numbers

- Sequence numbers for detecting message deletion, reordering and replay

  $A \rightarrow B$: B, seq, M, S_A(B, seq, M)

  Example:
  S_A("Transaction 43542. Transfer 30€ to account 1006443.")

Sequence number limitations

- Sequence number must grow monotonically
  - Difficult to implement in distributed endpoints, e.g. server farm, multi-threaded server
- Must not be reset, except when rekeying
- Sender and receiver counters must stay in sync
  - Plan resynchronization after message loss and endpoint failure
- Attacker can delay the message:

S_A(seq, “Bob, attack now!”)   // intercept and replay tomorrow

Nonces

- Checking freshness with B's nonce:
  1. A → B: "Hello, I'd like to send you a message."
  2. B → A: N_B
  3. A → B: $B, N_B, M, S_A(B, N_B, M)$
- Bob's nonce is usually a long random number selected by Bob
- Reasoning: any authenticated message that contains N_B must have been sent after Bob generated N_B

Nonce implementation

- Nonce must be never reused
- In many applications, nonce must be unpredictable to attackers
- Best nonce: 128-bit random number
  - Very unlikely to repeat and impossible to guess
- Another nonce: timestamp and random number (or their hash)
  - Protects against RNG problems, e.g., if entropy pool is empty after device reset

Nonce limitations

1. A → B: "Hello"
2. B → A: N_B
3. A → B: B, N_B, M, $S_A(B, N_B, M)$

- Nonce requires a random number generator, entropy source
- Nonce requires an extra message or roundtrip
- Ok for connections but not well suited for asynchronous communication, e.g., email, events, or message bus
- Not suitable for broadcast communication
  – Radio and satellite broadcast, multicast

Freshness mechanism summary

1. Use a random nonce from the receiver where possible
2. Timestamp to limit message lifetime + sequence number for duplicate detection
3. Use pure sequence number only when nothing else is available (leads to complex designs)