Announcements

• Project 3 is due next Monday, November 20th

• Plan for today:
  – Authentication protocols
Recap: Challenge Response

• Protocol doesn’t reveal the secret.

• Challenge/Response
  – Bart requests proof that Alice knows the secret
  – Alice requires proof from Bart
  – $R_A$ and $R_B$ are randomly generated numbers
Lessons

• Protocol design is tricky and subtle
  – “Optimizations” aren’t necessarily good
• Need to worry about:
  – Multiple instances of the same protocol running in parallel
  – Intruders that play by the rules, mostly
Threats

- **Transferability:** B cannot reuse an identification exchange with A to successfully impersonate A to a third party C.

- **Impersonation:** The probability is negligible that a party C distinct from A can carry out the protocol in the role of A and cause B to accept it as having A’s identity.
Assumptions

• A large number of previous authentications between A and B may have been observed.

• The adversary C has participated in previous protocol executions with A and/or B.

• Multiple instances of the protocol, possibly instantiated by C, may be run simultaneously.
Primary Attacks

- **Replay.**
  - Reusing messages (or parts of messages) inappropriately

- **Interleaving.**
  - Mixing messages from different runs of the protocol.

- **Reflection.**
  - Sending a message intended for destination A to B instead.

- **Chosen plaintext.**
  - Choosing the data to be encrypted

- **Forced delay.**
  - Denial of service attack -- taking a long time to respond
Primary Controls

• Replay:
  – use of challenge-response techniques
  – embed target identity in response.

• Interleaving
  – link messages in a session with chained nonces.

• Reflection:
  – embed identifier of target party in challenge response
  – use asymmetric message formats
  – use asymmetric keys.
Primary Controls, continued

• Chosen text:
  – embed self-chosen random numbers ("confounders") in responses
  – use "zero knowledge" techniques.

• Forced delays:
  – use nonces with short timeouts
  – use timestamps in addition to other techniques.
Replay

- *Replay*: the threat in which a transmission is observed by an eavesdropper who subsequently reuses it as part of a protocol, possibly to impersonate the original sender.
  - Example: Monitor the first part of a telnet session to obtain a sequence of transmissions sufficient to get a log-in.

- Three strategies for defeating replay attacks
  - Nonces
  - Timestamps
  - Sequence numbers.
Nonces: Random Numbers

- **Nonce**: A number chosen at random from a range of possible values.
  - Each generated nonce is valid only once.
- In a challenge-response protocol nonces are used as follows.
  - The verifier chooses a (new) random number and provides it to the claimant.
  - The claimant performs an operation on it showing knowledge of a secret.
  - This information is bound inseparably to the random number and returned to the verifier for examination.
  - A timeout period is used to ensure “freshness”.
Time Stamps

• The claimant sends a message with a timestamp.
• The verifier checks that it falls within an acceptance window of time.
• The last timestamp received is held, and identification requests with older timestamps are ignored.
• Good only if clock synchronization is close enough for acceptance window.
Sequence Numbers

• Sequence numbers provide a sequential or monotonic counter on messages.
• If a message is replayed and the original message was received, the replay will have an old or too-small sequence number and be discarded.
• Cannot detect forced delay.
• Difficult to maintain when there are system failures.
Unilateral Symmetric Key

- Unilateral = one way authentication
- Unilateral authentication with nonce.

\[ A \xrightarrow{K_{AB}\{n, B\}} \overleftarrow{n} \xleftarrow{K_{AB}\{n, B\}} B \]
Mutual Symmetric Key

- Mutual = two way authentication
- Using Nonces:

  $\text{Alice} \xrightarrow{\text{K}_{AB}\{\text{n}_{A}, \text{n}_{B}, \text{B}\}} \text{Bart}$

  $\text{Bart} \xleftarrow{\text{K}_{AB}\{\text{n}_{A}, \text{A}\}} \text{Alice}$

  $\text{Bart} \xrightarrow{\text{n}_{B}} \text{Alice}$

  $\text{Alice} \xleftarrow{\text{n}_{B}} \text{Bart}$
Mutual Public Key Decryption

- Exchange nonces

\[ K_B\{n_A, A\} \]

\[ K_A\{n_A, n_B\} \]

\[ n_B \]
Usurpation Attacks

- Identification protocols corroborate the identity of an entity only at a given instant in time.
  - An attacker could "hijack" a session after authentication.

- Techniques to assure ongoing authenticity:
  - Periodic re-identification.
  - Tying identification to an ongoing integrity service. For example: key establishment and encryption.
Multiple Use of Keys

- Risky to use keys for multiple purposes.
- Using an RSA key for both authentication and signatures may allow a chosen-text attack.
- B attacker/verifier, \( n_B = H(M) \) for some message \( M \).

B, pretending to be A
General Principles

• Don’t do anything more than necessary until confidence is built.
  – Initiator should prove identity before the responder does any “expensive” action (like encryption)

• Embed the intended recipient of the message in the message itself

• Principal that generates a nonce is the one that verifies it

• Before encrypting an untrusted message, add “salt” (i.e. a nonce) to prevent chosen plaintext attacks

• Use asymmetric message formats (either in “shape” or by using asymmetric keys) to make it harder for roles to be switched