Announcements

• Project 2 is due today at 11:59 pm.

• Midterm II
  – Thursday, March 22
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.
Mutual Symmetric Key

- Mutual = two way authentication
- Using Nonces:

\[ K_{AB}\{n_A, n_B, B\} \]

\[ K_{AB}\{n_A, A\} \]
Mutual Public Key Decryption

- Exchange nonces

\[ K_B\{n_A, A\} \quad \text{to} \quad K_A\{n_A, n_B\} \]

\[ n_B \]
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.
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
Physical Signatures

- Consider a paper check used to transfer money from one person to another
- Signature confirms authenticity
  - Only legitimate signer can produce signature
- In case of alleged forgery
  - 3rd party can verify authenticity
- Checks are cancelled
  - So they can’t be reused
- Checks are not alterable
  - Or alterations are easily detected
Digital Signatures: Requirements I

• A mark that only one principal can make, but others can easily recognize

• Unforgeable
  – If P signs a message M with signature $S_P\{M\}$ it is impossible for any other principal to produce the pair $(M, S_P\{M\})$.

• Authentic
  – If R receives the pair $(M, S_P\{M\})$ purportedly from P, R can check that the signature really is from P.
Digital Signatures: Requirements II

• Not alterable
  – After being transmitted, \((M, S_p\{M}\}) cannot be changed by P, R, or an interceptor.

• Not reusable
  – A duplicate message will be detected by the recipient.

• Nonrepudiation:
  – P should not be able to claim they didn't sign something when in fact they did.
  – (Related to unforgeability: If P can show that someone else could have forged P's signature, they can repudiate ("refuse to acknowledge") the validity of the signature.)
Digital Signatures with Shared Keys

Alice \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad

Tom is a trusted 3\textsuperscript{rd} party (or arbiter).

**Authenticity:** Tom verifies Alice’s message, Bart trusts Tom.

**No Forgery:** Bart can keep \texttt{msg}, \texttt{K\textsubscript{AT}\{msg\}}, which only Alice (or Tom, but he’s trusted not to) could produce
Preventing Reuse and Alteration

• To prevent reuse of the signature
  – Incorporate a *timestamp* (or sequence number)

• Alteration
  – If a block cipher is used, recipient could splice-together new messages from individual blocks.

• To prevent alteration
  – Timestamp must be part of each block
  – Or... use *cipher block chaining*
Digital Signatures with Public Keys

• Assumes the algorithm is commutative:
  – \( D(E(M, K), k) = E(D(M, k), K) \)
• Let \( K_A \) be Alice’s public key
• Let \( k_A \) be her private key
• To sign \( \text{msg} \), Alice sends \( D(\text{msg}, k_A) \)
• Bart can verify the message with Alice’s public key

• Works! RSA: \( (m^e)^d = m^{ed} = (m^d)^e \)
Digital Signatures with Public Keys

- No trusted 3rd party.
- Simpler algorithm.
- More expensive
- No confidentiality

Alice

\[ k_A \{ \text{msg} \} \]

Bart

\[ k_B, K_B, K_A \]

\( k_A, K_A, K_B \)
Variations on Public Key Signatures

• Timestamps again (to prevent replay)
  – Signed certificate valid for only some time.

• Add an extra layer of encryption to guarantee confidentiality
  – Alice sends $K_B \{k_A \{msg\}\}$ to Bart

• Combined with hashes:
  – Send $(msg, k_A \{MD5(msg)\})$
Unilateral Authentication: Signatures

• $S_A\{M\}$ is A’s signature on message M.
• Unilateral authentication with nonces:

The $n_A$ prevents chosen plaintext attacks.
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