6.033 Spring 2015
Lecture #23

• Combating network adversaries
  • Secure Channels
  • Signatures
**confidentiality:** adversary cannot learn message contents

**integrity:** adversary cannot tamper with message contents

(if they do, client and/or server will detect it)
encrypt(key, message) \rightarrow \text{ciphertext}

decrypt(key, ciphertext) \rightarrow \text{message}

**property:** given the \text{ciphertext}, it is (virtually) impossible to obtain the \text{message} without knowing the \text{key}

\text{MAC}(key, message) \rightarrow \text{token}

**property:** given the \text{message}, it is (virtually) impossible to obtain the \text{token} without knowing the \text{key}

(it is also impossible to go in the reverse direction)
alice

c = encrypt(k, m)
h = MAC(k, m)

bob

m = decrypt(k, c)
MAC(k, m) == h ?
problem: replay attacks
(adversary could intercept a message, re-send it at a later time)
\[
c = \text{encrypt}(k, m | \text{seq}) \\
h = \text{MAC}(k, m | \text{seq})
\]
problem: reflection attacks
(adversary could intercept a message, re-send it at a later time in the opposite direction)
alice

\[ c_a = \text{encrypt}(k_a, m_a \mid \text{seq}_a) \]
\[ h_a = \text{MAC}(k_a, m_a \mid \text{seq}_a) \]

bob

\[ m_a \mid \text{seq}_a = \text{decrypt}(k_a, c_a) \]
\[ \text{MAC}(k_a, m_a \mid \text{seq}_a) = h_a ? \]

\[ c_b = \text{encrypt}(k_b, m_b \mid \text{seq}_b) \]
\[ h_b = \text{MAC}(k_b, m_b \mid \text{seq}_b) \]

\[ m_b \mid \text{seq}_b = \text{decrypt}(k_b, c_b) \]
\[ \text{MAC}(k_b, m_b \mid \text{seq}_b) = h_b ? \]
problem: how do the parties know the keys?

known: $p$ (prime), $g$

property: given $g^r \mod p$, it is (virtually) impossible to determine $r$ even if you know $g$ and $p$

alice
pick random $a$

bob
pick random $b$

calculate $(g^b)^a \mod p$
calculate $(g^a)^b \mod p$

key = $g^{ab} \mod p$
**alice**
pick random \( a \)

\[
g^a \mod p \\
g^e \mod p
\]

\[
k_1 = (g^e)^a \mod p
\]

**eve**
pick random \( e \)

\[
g^e \mod p \\
g^b \mod p
\]

\[
k_2 = (g^e)^b \mod p
\]

eve can calculate \( k_1 \) and \( k_2 \)

**bob**
pick random \( b \)

\[
g^e \mod p \\
g^b \mod p
\]

\[
encrypt(k_2, m) \\
decrypt m
\]

**problem:** alice and bob don’t know they’re not communicating directly
cryptographic signatures

allow users to verify identities using public-key cryptography

\[
\text{sign}(\text{secret\_key}, \text{message}) \rightarrow \text{sig} \\
\text{verify}(\text{public\_key}, \text{message}, \text{sig}) \rightarrow \text{yes/no}
\]
TLS handshake

client

ClientHello \{version, \textit{seq}_c, session_id, cipher suites, compression func\}

ServerHello \{version, \textit{seq}_s, session_id, cipher suite, compression func\}

\{server certificate, CA certificates\}

ServerHelloDone

client verifies authenticity of server

ClientKeyExchange \{encrypt(server\_pub\_key, \textit{pre\_master\_secret})\}

compute

\texttt{master\_secret} = PRF(\textit{pre\_master\_secret}, “master secret”, \textit{seq}_c \mid \textit{seq}_s)

\texttt{key\_block} = PRF(\textit{master\_secret}, “key expansion”, \textit{seq}_c \mid \textit{seq}_s)

= \{\texttt{client\_MAC\_key},
        \texttt{server\_MAC\_key},
        \texttt{client\_encrypt\_key},
        \texttt{server\_encrypt\_key},
        ...
\}

Finished \{sign(\texttt{client\_MAC\_key}, encrypt(\texttt{client\_encrypt\_key},
                                               MAC(\textit{master\_secret}, \textit{previous\_messages}))))\}


server

Finished \{sign(\texttt{server\_MAC\_key}, encrypt(\texttt{server\_encrypt\_key},
                                               MAC(\textit{master\_secret}, \textit{previous\_messages}))))\}
• **Secure channels** protect us from adversaries that can observer and tamper with packets in the network.

• Encrypting with **symmetric keys** provides secrecy, and using **MACs** provides integrity. **Diffie-Hellman key exchange** lets us exchange the symmetric key securely.

• To verify identities, we use **public-key cryptography** and cryptographic **signatures**. We often distribute public keys with **certificate authorities**, though this method is not perfect.