Public key encryption

• Three randomized algorithms:
  - Generate – $G(1^k) \rightarrow K, K^{-1}$
  - Encrypt – $E(K, m) \rightarrow \{m\}_K$
  - Decrypt – $D(K^{-1}, \{m\}_K) \rightarrow m$

• Provides secrecy, like conventional encryption
  - Can’t derive $m$ from $\{m\}_K$ without knowing $K^{-1}$

• Encryption key $K$ can be made public
  - Can’t derive $K^{-1}$ from $K$
  - Everyone can use the same public key to encrypt messages for one recipient.
Digital signatures

• Three (randomized) algorithms:
  - Generate – $G(1^k) \rightarrow K, K^{-1}$
  - Sign – $S(K^{-1}, m) \rightarrow \{m\}_{K^{-1}}$
  - Verify – $V(K, \{m\}_{K^{-1}}, m) \rightarrow \{\text{true, false}\}$

• Provides integrity, like a MAC
  - Cannot produce valid $\langle m, \{m\}_{K^{-1}} \rangle$ pair without $K^{-1}$

• Many keys support both signing & encryption
  - But Encrypt/Decrypt and Sign/Verify different algorithms!
Cost of cryptographic operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>msec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt</td>
<td>1.11</td>
</tr>
<tr>
<td>Decrypt</td>
<td>39.62</td>
</tr>
<tr>
<td>Sign</td>
<td>40.56</td>
</tr>
<tr>
<td>Verify</td>
<td>0.10</td>
</tr>
</tbody>
</table>

[1,280-bit Rabin-Williams keys on 550 MHz K6]

- Cost of public key algorithms significant
  - Encryption only on small messages (< size of key)
  - Signature cost relatively insensitive to message size

- In contrast, symmetric algorithms must cheaper
  - Symmetric can encrypt+MAC faster than 100Mbit/sec LAN
Hybrid schemes

- **Use public key to encrypt symmetric key**
  - Send message symmetrically encrypted: \( \{ \text{msg} \}_{K_S}, \{ K_S \}_{K_P} \)

- **Use PK to negotiate secret session key**
  - E.g., Client sends server \( \{ K_1, K_2, K_3, K_4 \}_{K_P} \)
  - Client sends server: \( \{ m_1, \text{MAC}(K_2, m_1) \}_{K_1} \)
  - Server sends client: \( \{ m_2, \text{MAC}(K_4, m_2) \}_{K_3} \)

- **Often want mutual authentication (client & server)**
  - Or more complex, user(s), client, & server
Server authentication

• **An approach: Use public key cryptography**
  - Give client public key of server
  - Lets client authenticate secure channel to server

• **Problem: Key management problem**
  - How to get server’s public key?
  - How to know the key is really server’s?
The danger: Attackers impersonating servers

• File system example:
  - Attacker pretends to be server, gives its own public key
  - Attacker substitutes modified data for file
  - User writes sensitive file to fake server
Man in the middle attacks

- Attacker might not look like server
  - User would notice if file system didn’t contain right files

- Man in the middle attack foils user:
  - Attacker emulates server when talking to client
  - Attacker emulates client when talking to server
  - Attacker passes most messages through unmodified
  - Attacker substitutes own public key for client’s & server’s
  - Attacker records secret data, or tampers to cause damage
Key management

- Put public keys in the phone book
  - How do you know you have the real phone book?
  - How is a program supposed to use phone book www.phonebook.com? (are you talking to real web server)

- Exchange keys with people in person

- “Web of trust” – get keys from friends you trust
Certification authorities

- Everybody trusts some certification authority
- Everybody knows authority’s public key
  - E.g., built into web browser
Hierarchy with local trust

• To get from cs.nyu.edu to mit.edu:
  - cs.nyu.edu knows key for nyu.edu
  - nyu.edu knows key for edu/root
  - root knows key for mit.edu

• To get within cs.nyu.edu:
  - No need to trust outside authorities