Key Distribution and Exchange

What kind of keys do we share?

- Secret keys, of 1-key cryptography
- Public keys, of 2-key cryptography
Which do we prefer?

- For bulk encryption over a session
  - secret keys, for performance
- For authentication
  - public keys, for uncompromisability

Secure distribution of keys

- Public keys: trivial
- Secret keys: non-trivial
Public keys: distribution trivial

- Security doesn’t depend on public key
- Put them in a public database (DNS, phone book come to mind)

Secret keys: distribution options

- Physical delivery
  - A selects key, gives to B
  - C selects key, gives to A and B
- Data network delivery
  - A & B have a previous key, new key can be sent encrypted with old
  - A & B have encrypted connections to C, who selects and sends new key to A & B
- Last option, basis of Key Distribution Centers
Key Distribution Center

- Systems communicate with KDC using master keys
- Systems communicate with each other using transient session keys
- Systems get session keys from KDC

Scale, required keys for N hosts

- $N(N-1)/2$ session keys
  - the number of pairs of hosts
- $N$ master keys
  - the number of hosts
## Scale, required keys for N hosts

<table>
<thead>
<tr>
<th>No. of hosts</th>
<th>Session</th>
<th>Master</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N(N-1)/2</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>4,950</td>
<td>100</td>
</tr>
<tr>
<td>1000</td>
<td>499,500</td>
<td>1000</td>
</tr>
<tr>
<td>10000</td>
<td>49,995,000</td>
<td>10000</td>
</tr>
</tbody>
</table>

## Scale, required master keys

![Diagram showing the number of required master keys versus the number of hosts]
Scale, required session keys

Number of Required Session Keys

- Physical delivery, maybe feasible because
  - N relatively small
  - only done once
- Use public key system
  - KDC encrypts master key with stations’ respective public keys for delivery
KDC operation

- A wants to talk to B
- A asks KDC for a session key
- KDC generates one
- Using $K_B$ KDC encrypts
  - the session key
  - A’s address
- Using $K_A$ KDC encrypts
  - the above
  - the session key
- KDC sends whole package to A

KDC operation

- A decrypts received package
  - stores session key
  - sends B his portion
- B decrypts his portion
  - stores session key
  - corresponds it with the right party (A)
- Session ensues using session key
I prefer a self-securing connection, thank you

- A and B will negotiate their own key
- Without benefit of
  - a previous secret key between them
  - public and private key pairs belonging to them
  - an intermediate KDC
- In such manner that a free evesdropper cannot figure their key out

And free pie—
how ya gonna do *that*!!!

- It’s called Diffie-Hellman key exchange
- Allows an insecure channel to become secure
- There is information parties can exchange
  - that allows them to derive a common secret key
  - while disallowing an interceptor to do the same
“Primitive roots” of prime numbers

<table>
<thead>
<tr>
<th>n</th>
<th>$7^n$</th>
<th>$7^n \mod 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>49</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>343</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2,401</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>16,807</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>117,649</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>823,543</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>5,764,801</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>40,353,607</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>282,475,249</td>
<td>1</td>
</tr>
</tbody>
</table>

7 is a “primitive root” of 11 because

- For $n$ from 1 through 10
- The remainders of $7^n$ divided by 11
- Consist of the numbers 1 through 10
Diffie-Hellman operation

- A picks a prime p and its primitive root g
- A sends p and g to B
- A picks a random integer less than p
- B picks a random integer less than p
- A calculates the remainder, when divided by p, of g raised to his integer
- B calculates the remainder, when divided by p, of g raised to his integer

Diffie-Hellman operation

- A sends his remainder to B
- B sends his remainder to A
- A calculates the remainder, when divided by p, of B’s remainder raised to A’s integer
- B calculates the remainder, when divided by p, of A’s remainder raised to B’s integer
- The results are always the same number
Diffie-Hellman example

- A picks a prime 11 and its primitive root 7
- A sends 11 and 7 to B
- A picks integer 3
- B picks integer 6
- A calculates the remainder, when divided by 11, of 7 raised to power 3 (it’s 2)
- B calculates the remainder, when divided by 11, of 7 raised to power 6 (it’s 4)

textbook, pp. 96-97

Diffie-Hellman example

- A sends 2 to B
- B sends 4 to A
- A calculates the remainder, when divided by 11, of 4 raised to power 3
- B calculates the remainder, when divided by 11, of 2 raised to power 6
- Both results are 9, the new secret key
Interceptor resistance

- Interceptor gets p and g, and both parties’ remainders
- Interceptor doesn’t get parties’ integers
- The integers are needed to calculate the key
- Deriving the a party’s integer from his remainder is mathematically infeasible for large p