Symmetric Cryptography

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials...
Goals for Today

- Cryptography

- Also: Lab part 1 due on Friday
  - Don’t all increase in complexity
  - Read recommended readings
One-Time Pad

Key is a random bit sequence as long as the plaintext

Encrypt by bitwise XOR of plaintext and key:
plaintext \oplus key

Decrypt by bitwise XOR of ciphertext and key:
ciphertext \oplus key =
(plaintext \oplus key) \oplus key =
plaintext \oplus (key \oplus key) =
plaintext
Advantages of One-Time Pad

- Easy to compute
  - Encryption and decryption are the same operation
  - Bitwise XOR is very cheap to compute

- As secure as theoretically possible
  - Given a ciphertext, all plaintexts are equally likely, regardless of attacker’s computational resources
  - ...as long as the key sequence is truly random
    - True randomness is expensive to obtain in large quantities
  - ...as long as each key is same length as plaintext
    - But how does the sender communicate the key to receiver?
Disadvantages

Key is a random bit sequence as long as the plaintext

Encrypt by bitwise XOR of plaintext and key:
\[ \text{ciphertext} = \text{plaintext} \oplus \text{key} \]

Decrypt by bitwise XOR of ciphertext and key:
\[ \text{ciphertext} \oplus \text{key} = (\text{plaintext} \oplus \text{key}) \oplus \text{key} = \text{plaintext} \oplus (\text{key} \oplus \text{key}) = \text{plaintext} \]

Disadvantage #1: Keys as long as messages.
Impractical in most scenarios
Still used by intelligence communities
Disadvantages

Key is a random bit sequence as long as the plaintext

Encrypt by bitwise XOR of plaintext and key:
\[ \text{ciphertext} = \text{plaintext} \oplus \text{key} \]

Decrypt by bitwise XOR of ciphertext and key:
\[ \text{ciphertext} \oplus \text{key} = (\text{plaintext} \oplus \text{key}) \oplus \text{key} = \text{plaintext} \]

Disadvantage #2: No integrity protection
Disadvantages

Disadvantage #3: Keys cannot be reused

Learn relationship between plaintexts:

\[ C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = (P_1 \oplus P_2) \oplus (K \oplus K) = P_1 \oplus P_2 \]
Visual Cryptography

• Generate a random bitmap

• Encode 0 as:

• Encode 1 as:
Visual Cryptography

- Take a black and white bitmap image
- For a white pixel, send the same as the mask
  
  or

- For a black pixel, send the opposite of the mask
  
  
See also http://www.cs.washington.edu/homes/yoshi/cs4hs/cse-vc.html
Visual Cryptography

- http://www.cl.cam.ac.uk/~fms27/vck/face.gif

See also http://www.cs.washington.edu/homes/yoshi/cs4hs/cse-vc.html
Reducing Keysize

- What do we do when we can’t pre-share huge keys?
  - When OTP is unrealistic

- We use special cryptographic primitives
  - Single key can be reused (with some restrictions)
  - But no longer provable secure (in the sense of the OTP)

- Examples: Block ciphers, stream ciphers
Background: Permutation

- For N-bit input, $2^N!$ possible permutations
- Idea for how to use a **keyed** permutation: split plaintext into blocks; for each block use **secret key** to pick a permutation
  - Without the key, permutation should “look random”
Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Each key defines a different permutation
  - Same key is reused for each block (can use short keys)
Block Cipher Security

- Result should look like a random permutation on the inputs
  - Recall: not just shuffling bits. N-bit block cipher permutes over $2^N$ inputs.

- Only computational guarantee of secrecy
  - Not impossible to break, just very expensive
    - If there is no efficient algorithm (unproven assumption!), then can only break by brute-force, try-every-possible-key search
  - Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information
Block Cipher Operation (Simplified)

Block of plaintext

S
S
S
S

Add some secret key bits to provide confusion

Key

Each S-box transforms its input bits in a “random-looking” way to provide diffusion (spread plaintext bits throughout ciphertext)

repeat for several rounds

Block of ciphertext

Procedure must be reversible (for decryption)
Feistel Structure (Stallings Fig 2.2)
**DES**

- **Feistel structure**
  - “Ladder” structure: split input in half, put one half through the round and XOR with the other half
  - After 3 random rounds, ciphertext indistinguishable from a random permutation if internal F function is a pseudorandom function (Luby & Rackoff)

- **DES: Data Encryption Standard**
  - Feistel structure
  - Invented by IBM, issued as federal standard in 1977
  - 64-bit blocks, 56-bit key + 8 bits for parity
DES and 56 bit keys (Stallings Tab 2.2)

56 bit keys are quite short

<table>
<thead>
<tr>
<th>Key Size (bits)</th>
<th>Number of Alternative Keys</th>
<th>Time required at 1 encryption/μs</th>
<th>Time required at 10^6 encryptions/μs</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>$2^{32} = 4.3 \times 10^9$</td>
<td>$2^{31} \mu s = 35.8$ minutes</td>
<td>2.15 milliseconds</td>
</tr>
<tr>
<td>56</td>
<td>$2^{56} = 7.2 \times 10^{16}$</td>
<td>$2^{55} \mu s = 1142$ years</td>
<td>10.01 hours</td>
</tr>
<tr>
<td>128</td>
<td>$2^{128} = 3.4 \times 10^{38}$</td>
<td>$2^{127} \mu s = 5.4 \times 10^{24}$ years</td>
<td>5.4 x 10^{18} years</td>
</tr>
<tr>
<td>168</td>
<td>$2^{168} = 3.7 \times 10^{50}$</td>
<td>$2^{167} \mu s = 5.9 \times 10^{36}$ years</td>
<td>5.9 x 10^{30} years</td>
</tr>
<tr>
<td>26 characters</td>
<td>$26! = 4 \times 10^{26}$</td>
<td>$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years</td>
<td>6.4 x 10^{6} years</td>
</tr>
</tbody>
</table>

1999: EFF DES Crack + distributed machines
- < 24 hours to find DES key

DES ---> 3DES
- 3DES: DES + inverse DES + DES (with 2 or 3 diff keys)
Advanced Encryption Standard (AES)

- New federal standard as of 2001
- Based on the Rijndael algorithm
- 128-bit blocks, keys can be 128, 192 or 256 bits
- Unlike DES, does not use Feistel structure
  - The entire block is processed during each round
- Design uses some very nice mathematics
Basic Structure of Rijndael

128-bit plaintext
(arranged as 4x4 array of 8-bit bytes)

⊕

S
byte substitution

Shift rows
(1\textsuperscript{st} unchanged, 2\textsuperscript{nd} left by 1, 3\textsuperscript{rd} left by 2, 4\textsuperscript{th} left by 3)

Mix columns
(each new byte depends on all bytes in old column)

add key for this round

repeat 10 times

128-bit key

Expand key

Mix columns

Shift rows

byte substitution

128-bit plaintext
(arranged as 4x4 array of 8-bit bytes)
Encrypting a Large Message

So, we’ve got a good block cipher, but our plaintext is larger than 128-bit block size

128-bit plaintext
(arranged as 4x4 array of 8-bit bytes)

128-bit ciphertext

What should we do?
Electronic Code Book (ECB) Mode

- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks