Symmetric Cryptography

Spring 2021

Tadayoshi Kohno
yoshi@cs
Admin

• Lab 1 Checkpoint: Today
  • Additional TA office hours offered today

• Homework 2: Out ~early next week
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)
### Keyed Permutation

For N-bit input, $2^N!$ possible permutations
For K-bit key, $2^K$ possible keys

<table>
<thead>
<tr>
<th>input</th>
<th>possible output (K=00)</th>
<th>possible output (K=01)</th>
<th>etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>010</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>111</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>101</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>110</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>000</td>
<td>110</td>
<td></td>
</tr>
</tbody>
</table>
Keyed Permutation

- Not just shuffling of input bits!
  - Suppose plaintext = “111”.
  - Then “111” is **not** the only possible ciphertext!
- Instead:
  - **Permutation of possible outputs**
  - Use secret key to pick a permutation
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
  • “Break” could mean recovering key, or it could mean distinguishing the block cipher’s behavior from that of a randomly selected permutation over the $2^N$ possible inputs
New Block Cipher Slides
Block Cipher Operation (Simplified)

Add some secret key bits to provide confusion.

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

Procedure must be reversible (for decryption).
Standard Block Ciphers

• **DES: Data Encryption Standard**
  • Feistel structure: builds invertible function using non-invertible ones
  • Invented by IBM, issued as federal standard in 1977
  • 64-bit blocks, 56-bit key + 8 bits for parity
DES and 56 bit keys

- 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>$5.4 \times 10^{18}$ 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>$5.9 \times 10^{30}$ years</td>
</tr>
<tr>
<td>26 characters (permutation)</td>
<td>$26! = 4 \times 10^{26}$</td>
<td>$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years</td>
<td>$6.4 \times 10^6$ 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)
3DES

• Two-key 3DES increases security of DES by doubling the key length
But wait... what about 2DES?

• Suppose you are given plaintext-ciphertext pairs (P1,C1), (P2,C2), (P3,C3)
• Suppose Key1 and Key2 are each 56-bits long
• Can you figure out Key1 and Key2 if you try all possible values for both ($2^{112}$ possibilities) → Yes
• Can you figure out Key1 and Key2 more than that? → Breakout
Meet-in-the-Middle Attack

• Guess $2^{56}$ values for Key1, and create a table from P1 to a middle value M1 for each key guess ($M1^{G1}$, $M1^{G2}$, $M1^{G3}$, ...)

• Guess $2^{56}$ values for Key2, and create a table from C1 to a middle value M’1 for each key guess ($M’1^{G1}$, $M’1^{G2}$, $M’1^{G3}$, ...)

• Look for collision in the middle values → if only one collision, found Key1 and Key2; otherwise repeat for (P2,C2), ...
Defining the strength of a scheme

- **Effective Key Strength**
  - Amount of ‘work’ the adversary needs to do

- **DES**: 56-bits
  - $2^{56}$ encryptions to try ‘all keys’

- **2DES**: 57-bits
  - $2 \times 2^{56}$ encryptions = $2^{57}$

- **3DES**: 112-bits (or sometimes 80-bits)
  - Meet-in-the-middle + more work = $2^{112}$ (for 3 keys, e.g. K1, K2, K3)
  - Various attacks = $2^{80}$ (for 2 keys, e.g. K1, K2, K1)
Standard Block Ciphers

• **DES: Data Encryption Standard**
  • Feistel structure: builds invertible function using non-invertible ones
  • Invented by IBM, issued as federal standard in 1977
  • 64-bit blocks, 56-bit key + 8 bits for parity

• **AES: Advanced Encryption Standard**
  • New federal standard as of 2001
    • NIST: National Institute of Standards & Technology
  • Based on the Rijndael algorithm
    • Selected via an open process
  • 128-bit blocks, keys can be 128, 192 or 256 bits
Encrypting a Large Message

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

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

- Plaintext message is broken into blocks (e.g., 128-bit blocks if the block cipher operates on 128-bit inputs).
- Each block is encrypted with a block cipher with the key.
- The ciphertext is the concatenation of each block cipher output.
Breakout

• What security concerns do you see with the ECB ("electronic code book") block cipher mode?

- Plaintext message is broken into blocks (e.g., 128-bit blocks if the block cipher operates on 128-bit inputs).
- Each block is encrypted with a block cipher with the key.
- The ciphertext is the concatenation of each block cipher output.
Electronic Code Book (ECB) Mode

- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks
Information Leakage in ECB Mode

[Wiki:**edia]

Encrypt in ECB mode
Move Fast and Roll Your Own Crypto
A Quick Look at the Confidentiality of Zoom Meetings

By Bill Marczak and John Scott-Railton    April 3, 2020

- Zoom documentation claims that the app uses “AES-256” encryption for meetings where possible. However, we find that in each Zoom meeting, a single AES-128 key is used in ECB mode by all participants to encrypt and decrypt audio and video. The use of ECB mode is not recommended because patterns present in the plaintext are preserved during encryption.

Cipher Block Chaining (CBC) Mode: Encryption

- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity
CBC Mode: Decryption

plaintext

Initialization vector

key

Decryption process:
- Initialization vector is added to the plaintext.
- The result is then decrypted with the key.
- The same process is repeated for each block of ciphertext.

Ciphertext
ECB vs. CBC

AES in ECB mode

Similar plaintext blocks produce similar ciphertext blocks (not good!)

AES in CBC mode

[Picture due to Bart Preneel]
Initialization Vector Dangers

Found in the source code for Diebold voting machines:

```
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data, totalSize, DESKEY, NULL, DES_ENCRYPT)
```
Counter Mode (CTR): Encryption

- Identical blocks of plaintext encrypted differently
- Still does not guarantee integrity; Fragile if "ctr" repeats
Counter Mode (CTR): Decryption

\[
\begin{align*}
\text{ct}_0 & \quad + \quad \text{key} \quad \text{block cipher} \\
\text{ct}_1 & \quad + \quad \text{key} \quad \text{block cipher} \\
\text{ct}_2 & \quad + \quad \text{key} \quad \text{block cipher} \\
\text{ct}_3 & \quad + \quad \text{key} \quad \text{block cipher} \\
\text{ct}_4 & \quad + \quad \text{key} \\
\text{pt}_1 & \quad + \quad \text{ct}_0 \\
\text{pt}_2 & \quad + \quad \text{ct}_1 \\
\text{pt}_3 & \quad + \quad \text{ct}_2 \\
\text{pt}_4 & \quad + \quad \text{ct}_3
\end{align*}
\]