CSE 484 / CSE M 584: Symmetric Encryption

Fall 2023

Franziska (Franzi) Roesner
franzi@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Announcements

• Homework 2 (crypto) to be released soon
  – Due ~1 week after Lab 1b
  – Designed to give you hands-on experience with crypto concepts, not be tricky
Reminder: Reducing Key Size

• What to do when it is infeasible to pre-share huge random keys?
  – When one-time pad is unrealistic...

• Use special cryptographic primitives: block ciphers, stream ciphers
  – Single key can be re-used (with some restrictions)
  – Not as theoretically secure as one-time pad
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</th>
<th>possible output</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

• 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)

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)
Two-key 3DES increases security of DES by doubling the key length.

Why not 2DES?

- Minimal gain in security compared to DES due to meet-in-the-middle attack (more in section on this)
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

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

128-bit ciphertext

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

Canvas “quiz” time!
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

[Encrypt in ECB mode]

[Wikipedia]
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](https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/) 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

decrypt

ciphertext
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:

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

- Initial ctr (random) + Key → ctr
- ctr +1 + Key → ctr+1
- ctr+2 + Key → ctr+2
- ctr+3 + Key → ctr+3

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

Initial ctr → ctr → ctr+1 → ctr+2 → ctr+3

Key → block cipher → Key → block cipher → Key → block cipher → Key → block cipher

ct ⊕ ct ⊕ ct ⊕ ct → pt → pt → pt → pt