Recap: 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)
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)
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

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

[Encrypted Text]

[Image: Tux (Linux mascot) on the left and a pixelated image on the right, connected with an arrow labeled 'Encrypt in ECB mode'.]
Cipher Block Chaining (CBC) Mode: Encryption

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

Initialization vector (random)

Sent with ciphertext (preferably encrypted)
CBC Mode: Decryption

Initialization vector

plaintext

ciphertext

decrypt
decrypt
decrypt
decrypt

key

key

key

key

10/18/17
AES in ECB mode

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

AES in CBC mode

[Picture due to Bart Preneel]
Initialization vector (supposed to be random)

CBC and Electronic Voting

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

Initial ctr

\[ \text{ctr} \rightarrow \text{ctr+1} \rightarrow \text{ctr+2} \rightarrow \text{ctr+3} \]

\[ \text{Key} \rightarrow \text{block cipher} \rightarrow \text{ct} \rightarrow \text{pt} \]

\[ \text{Key} \rightarrow \text{block cipher} \rightarrow \text{ct} \rightarrow \text{pt} \]

\[ \text{Key} \rightarrow \text{block cipher} \rightarrow \text{ct} \rightarrow \text{pt} \]

\[ \text{Key} \rightarrow \text{block cipher} \rightarrow \text{ct} \rightarrow \text{pt} \]
When is an Encryption Scheme “Secure”?

• Hard to recover the key?
  – What if attacker can learn plaintext without learning the key?

• Hard to recover plaintext from ciphertext?
  – What if attacker learns some bits or some function of bits?
How Can a Cipher Be Attacked?

• Attackers knows ciphertext and encryption algthm
  – What else does the attacker know? Depends on the application in which the cipher is used!

• Ciphertext-only attack

• KPA: Known-plaintext attack (stronger)
  – Knows some plaintext-ciphertext pairs

• CPA: Chosen-plaintext attack (even stronger)
  – Can obtain ciphertext for any plaintext of his choice

• CCA: Chosen-ciphertext attack (very strong)
  – Can decrypt any ciphertext except the target
Chosen Plaintext Attack

Crook #1 changes his PIN to a number of his choice

PIN is encrypted and transmitted to bank

cipher(key,PIN)

Crook #2 eavesdrops on the wire and learns ciphertext corresponding to chosen plaintext PIN

... repeat for any PIN value
Very Informal Intuition

• Security against chosen-plaintext attack (CPA)
  – Ciphertext leaks no information about the plaintext
  – Even if the attacker correctly guesses the plaintext, he cannot verify his guess
  – Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts
  • Implication: encryption must be randomized or stateful

• Security against chosen-ciphertext attack (CCA)
  – Integrity protection – it is not possible to change the plaintext by modifying the ciphertext