Block Cipher

Encrypting a large message

**Electronic Code Book (ECB)**

![Diagram showing the encryption process in ECB mode]
Block Cipher

Decrypting a large message

Electronic Code Book (ECB)
Block Cipher

En/Decrypting a large message

**Electronic Code Book (ECB)**

**Problems:**

Two same message blocks encrypt to the same cipher blocks

1. Two cipher blocks can be switched
2. One cipher block can be copied to another
   
   ex: switch or copy salary block
3. No built-in integrity or authentication check

Possible fix: have many keys, one for each block

Recurring phrases cause repeated part-blocks of ciphertext

Plaintext patterns become obvious under codebook attack

If attacker can dupe sender into sending known plaintext...

Possible fix: send large blocks and add random bits to each
Encrypting a large message

Cipher Block Chaining (CBC) – 1\textsuperscript{st} attempt  

r1...r6 are random

Block Cipher
Block Cipher

En/Decrypting a large message

Cipher Block Chaining (CBC) - 1st attempt

Problems:

1. Not efficient – one random number for every message block
2. Attacker can rearrange blocks with predictable effect on resulting plaintext. For example, just remove one block or swap two blocks - result can still be decrypted and receiver does not know the difference.
3. If an attacker knows the value of any message block $m_i$, then can change it in a predictable way by modifying $r_i$.

Since random $r_i$ are sent with the message, attacker can modify them
Block Cipher

Encrypting a large message

**Cipher Block Chaining (CBC)**  IV is a random number

```
message

m1 m2 m3 m4 m5 m6

c1 c2 c3 c4 c5 c6
```

Secret $\oplus$ $\oplus$ $\oplus$ $\oplus$ $\oplus$ $\oplus$

Secret $\rightarrow$ E $\rightarrow$ E $\rightarrow$ E $\rightarrow$ E $\rightarrow$ E $\rightarrow$ E

IV $\rightarrow$ $\oplus$ $\oplus$ $\oplus$ $\oplus$ $\oplus$ $\oplus$
Block Cipher

Decrypting a large message

Cipher Block Chaining (CBC)  IV is a random number

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
<th>m6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c2</td>
<td>c3</td>
<td>c4</td>
<td>c5</td>
<td>c6</td>
</tr>
</tbody>
</table>

Secret ⊕ D ⊕ D ⊕ D ⊕ D ⊕ D

IV ⊕ m1 ⊕ m2 ⊕ m3 ⊕ m4 ⊕ m5 ⊕ m6
Block Cipher

En/Decrypting a large message

Cipher Block Chaining (CBC)

Discussion:
1. Must use random IV – guarantees that same plaintext causes different ciphertext
   If IV is not random, information is revealed even if message not decrypted

Examples:
   commander orders troops to hold several times then attack
   If salary fields are known, can determine whose salary has changed

Benefit:
   attackers cannot supply chosen plaintext to the encryption algorithm itself, even if chosen plaintext can be supplied to the CBC
Block Cipher

En/Decrypting a large message

Cipher Block Chaining (CBC)

Discussion:

1. Must use random IV – guarantees that same plaintext causes different ciphertext
   If IV is not random, information is revealed even if message not decrypted

Examples:
   commander orders troops to hold several times then attack
   If salary fields are known, can determine whose salary has changed

Benefit:
   attackers cannot supply chosen plaintext to the encryption algorithm
   Itself, even if chosen plaintext can be supplied to the CBC

2. Attacker can rearrange blocks with predictable effect on resulting plaintext.
   Changing $c_i$ has a predictable effect on $m_{i+1}$. Might decrypt to this:

   Hello 7834  →  &8*# 7835
Block Cipher

En/Decrypting a large message

Cipher Block Chaining (CBC)

Discussion:
1. Must use random IV – guarantees that same plaintext causes different ciphertext
   If IV is not random, information is revealed even if message not decrypted
   Examples:
   commander orders troops to hold several times then attack
   If salary fields are known, can determine whose salary has changed

   Benefit:
   attackers cannot supply chosen plaintext to the encryption algorithm
   Itself, even if chosen plaintext can be supplied to the CBC

2. Attacker can rearrange blocks with predictable effect on resulting plaintext.
   Changing $c_i$ has a predictable effect on $m_{i+1}$. Might decrypt to this:

   \[
   \begin{array}{c|c}
   \text{Hello} & 7834 \\
   \end{array}
   \rightarrow
   \begin{array}{c|c}
   & 8*\# \\
   \end{array}
   \begin{array}{c|c}
   & 7835 \\
   \end{array}
   \]

3. If $m_1...m_n$ and $c_1...c_n$ and IV are known, all decryptions of $c_i$ are known.
   If enough of these are obtained, a new ciphertext can be constructed and
   the decrypt would be known.
Entropy

Suppose the set of characters I transmit is \{A\} (i.e. one character). What is the probability that the next character in a transmission stream is the character A?
Block Cipher

Entropy

Suppose the set of characters I transmit is \{A\} (i.e. one character) What is the probability that the next character in a transmission stream is the character A?

Answer: 1 (we have complete predictability)
Entropy

Suppose the set of characters I transmit is \{A, B\} (i.e. two characters). What is the probability that the next character in a transmission stream is the character A if the stream characters are randomly selected for insertion?
Block Cipher

Entropy

Suppose the set of characters I transmit is \{A,B\} (i.e. two characters)
What is the probability that the next character in a transmission stream is the character A if the stream characters are randomly selected for insertion?

Answer: $\frac{1}{2}$ (complete uncertainly)
Entropy

Suppose the set of characters I transmit is \{A,B\} (i.e. two characters) What is the probability that the next character in a transmission stream is the character A if the stream characters are randomly selected for insertion?

Answer: \( \frac{1}{2} \) (complete uncertainly)

If there are \( n \) characters inserted randomly in a stream
\[ \Pr(\text{next one is A}) = \frac{1}{n} \]
Entropy

Suppose the set of characters I transmit is \{A,B\} (i.e. two characters) What is the probability that the next character in a transmission stream is the character A if the stream characters are randomly selected for insertion?

Answer: \(\frac{1}{2}\) (complete uncertainly)

If there are \(n\) characters inserted randomly in a stream
\[\text{Pr(next one is A)} = \frac{1}{n}\]

Entropy expresses the minimum number of bits needed to encode a sequence of symbols.

Example: characters \{A,B,C,D\},
\[\text{Pr}(A) = \frac{1}{2}, \text{Pr}(B)=\frac{1}{4} , \text{Pr}(C) = \text{Pr}(D) = \frac{1}{8}\]
Let 1 be transmission of A, 01 be trans of B, 001 be C, 010 be D
Entropy of a Symbol

Given alphabet $S=\{s_1, s_2, \ldots, s_n\}$ with probs $\{p_1, p_2, \ldots, p_n\}$ of occurring in message $M$ of length $m$. Define entropy

$$H(S) = -\sum_i p_i \log_2(p_i) \quad \text{for all non-zero } p_i$$

Observe: if all $p_i$ are equal, $H(S) = \log_2(n)$

- if $p_1=1$, all other $p_i=0$, $H(S) = 0$.
- if $p_1 = \frac{1}{2}$ and $p_2 = \frac{1}{2}$, other $p_i=0$, $H(S)=1$.

$\frac{1}{2}$ bit of entropy if $p_1 = 0.11002786..$, $p_2 = 0.88997213...$

If all probabilities are equal,

$$\Pr(\text{next character is } A \mid \text{prev char}) = \Pr(\text{next character is } A)$$

The higher the entropy the more secure the cryptosystem is
Block Cipher

Entropy and the xor operation

Let $p_1$ be the probability that 1 is the next message bit $m_i$ and $p_0$ is the probability that 0 is the next $m_i$.

Then $\Pr (m_1 \oplus r_1 \text{ is 0}) = (\frac{1}{2})p_1 + (\frac{1}{2})p_0 = \frac{1}{2}$

$\Pr (m_1 \oplus r_1 \text{ is 1}) = (\frac{1}{2})p_0 + (\frac{1}{2})p_1 = \frac{1}{2}$

$H(c_i) = 1$ \hspace{1em} \textbf{Unconditionally secure}: $H(m_i | c_i) = H(m_i)$

Regardless of correlations in the message bits, the xor operation gives the highest entropy and greatest security!
Encrypting a large message

**Output Feedback Mode (OFB)** IV is a random number

Block Cipher

\[ m_1 \oplus c_1 = k \text{ bits} \]
\[ m_2 \oplus c_2 = k \text{ bits} \]
\[ m_3 \oplus c_3 = k \text{ bits} \]
En/Decrypting a large message

Output Feedback Mode vs Cyber Block Chaining

Discussion:

1. OFB: one-time pad can be generated in advance, encryption is based solely on (cheap) exclusion-or operation

2. OFB: garbled cipher block affects only its corresponding message block
   CBC: garbled cipher block affects two message blocks

3. OFB: portions of message can be encrypted and sent as bytes arrive
   CBC: must wait for a block to arrive before encrypting

4. OFB: if the plaintext and ciphertext are known by attacker, plaintext can be modified to anything by xoring ciphertext with the known plaintext

5. OFB and CBC: if any character is lost in transmission, rest of output may be garbled unless some sync markers are added
Block Cipher

Encrypting a large message

**Cipher Feedback Mode (CFB)** IV is a random number

```
IV
K → E
discard
m1 ⊕ k bits
→ c1 k bits

K → E
discard
m2 ⊕ k bits
→ c2

K → E
discard
m3 ⊕ k bits
→ c3
```
Block Cipher

En/Decrypting a large message

**Output Feedback Mode vs Cipher Feedback Mode**

**Discussion:**

1. In OFB one-time pad can be generated before message is. Not so for CFB

2. In 8-bit CFB loss of bytes in transmission will synchronize after pad flushes through shift. Added bytes will also synchronize after and extra plaintext byte plus 8 garbage bytes. Not so for OFB or CBC where rest of transmission is garbled.

3. No block rearrangement attack on CFB although sections can be rearranged at the cost of garbling the splice points.

4. CFB: one DES operation for every byte of ciphertext (costly)
Encrypting a large message

**Counter Mode (CTR)**

```
IV
K → E
m1 → ⊕
c1

IV+1
K → E
m2 → ⊕
c2

IV+2
K → E
m3 → ⊕
c3
```
En/Decrypting a large message

**Counter Mode (CTR)**

**Discussion:**

1. Like OFB, one-time pad is generated before the message is. Encryption is simple with exclusive-or.

2. Like CBC, can decrypt beginning from any point in the ciphertext. Useful for encrypting random access files.

3. If different data is used with same key and IV, exclusive-oring the ciphertexts of the messages gives the exclusive-or of the plaintexts. This is also a problem with OFB.
Generating Message Integrity Check (MIC)

Suppose message is sent in the clear

Only send the residue as the check on the ciphertext and the plaintext message (no confidentiality)
Generating Message Integrity Check (MIC)

Integrity plus confidentiality

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
<th>m6</th>
</tr>
</thead>
</table>

IV → ⊕ → ⊕ → ⊕ → ⊕ → ⊕ → ⊕

K → E → E → E → E → E → E

<table>
<thead>
<tr>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
</tr>
</thead>
</table>

CBC residue
Generating Message Integrity Check (MIC)

Integrity plus confidentiality

Huh? Send last block twice? Tamperer merely sends tampered message and just repeats its last block!!
Generating Message Integrity Check (MIC)

Integrity plus confidentiality

c6 is the residue.
Generating Message Integrity Check (MIC)

Integrity plus confidentiality

\[
\begin{array}{cccccc}
  m1 & m2 & m3 & m4 & m5 & c5 \\
\end{array}
\]

\[
\begin{array}{cccccc}
  c1 & c2 & c3 & c4 & c5 & c6 \\
\end{array}
\]

\[
\begin{align*}
  IV & \oplus & E \\
  K & \rightarrow & E & \rightarrow & E & \rightarrow & E & \rightarrow & E & \rightarrow & E \\
\end{align*}
\]

c6 is the residue. But actually c6 is 0!!
Generating Message Integrity Check (MIC)

Cyclic redundancy check

11010011101100 000  Data + padding
1011  Divisor
-----------------------
01100011101100 000  Result
Generating Message Integrity Check (MIC)

Cyclic redundancy check

\[ 11010011101100 \ 000 \]
\[ 1011 \]
\[ \underline{1011} \]
\[ 01100011101100 \ 000 \quad \text{Modified data} \]
\[ 1011 \quad \text{Divisor} \]
\[ \underline{1011} \]
\[ 00111011101100 \ 000 \quad \text{Result} \]
Generating Message Integrity Check (MIC)

Cyclic redundancy check

11010011101100 000
1011

------------------------
01100011101100 000
1011

------------------------
00111011101100 000 Modified data
1011 Divisor

------------------------
00010111101100 000 Result
Generating Message Integrity Check (MIC)

Cyclic redundancy check

\[
\begin{array}{c}
11010011101100 \ 000 \\
1011 \\
\hline
01100011101100 \ 000 \\
1011 \\
\hline
00111011101100 \ 000 \\
1011 \\
\hline
00010111101100 \ 000 \\
1011 \\
\hline
00000001101100 \ 000
\end{array}
\]

Modified data

Divisor

Result
Generating Message Integrity Check (MIC)

Cyclic redundancy check

```
11010011101100 000
1011
-----------------------
01100011101100 000
1011
-----------------------
00111011101100 000
1011
-----------------------
00010111101100 000
1011
-----------------------
00000001101100 000
```

Modified data
Divisor

```
00000000110100 000
```

Result
Generating Message Integrity Check (MIC)

Cyclic redundancy check

--------------------------
000000000110100 000
1011
--------------------------
00000000011000 000
  Modified data
  Divisor
--------------------------
00000000011000 000
  Result
Generating Message Integrity Check (MIC)

Cyclic redundancy check

```
00000000110100 000
  1011
---------------------
00000000011000 000       Modified data
  1011            Divisor
---------------------
00000000001110 000       Result
```
Generating Message Integrity Check (MIC)

Cyclic redundancy check

```
00000000110100 000
  1011

00000000011000 000
  1011

00000000011100 000
  1011

00000000001110 000
  Modified data
  1011
  Divisor

00000000000101 000
  Result
```
Generating Message Integrity Check (MIC)

Cyclic redundancy check

\[
\begin{array}{c}
00000000110100 000 \\
1011 \\
\hline
00000000011000 000 \\
1011 \\
\hline
00000000001110 000 \\
1011 \\
\hline
00000000000101 000 \\
101 100 \\
\hline
00000000000000 100 \\
\end{array}
\]

Result  Divisor  CRC
Generating Message Integrity Check (MIC)

Integrity plus confidentiality

c6 is the residue. CRC is used.
Generating Message Integrity Check (MIC)

To use CBC for both message integrity and encryption, use different keys for the residue and ciphertext!