# Block Cipher Modes

## Overview

### CBC

\[
\begin{align*}
C_0 &= \text{IV} \\
C_j &= E_K(C_{j-1} \oplus P_j) \\
P_1 &= D_K(C_1) \oplus \text{IV} \\
P_j &= D_K(C_j) \oplus C_{j-1}
\end{align*}
\]

### CFB

\[
\begin{align*}
C_0 &= \text{IV} \\
C_j &= E_K(C_{j-1} \oplus P_j) \\
P_1 &= E_K(\text{IV}) \oplus C_1 \\
P_j &= E_K(C_{j-1}) \oplus C_j
\end{align*}
\]

### OFB

\[
\begin{align*}
O_0 &= \text{IV} \\
O_j &= E_K(O_{j-1}) \\
C_j &= O_j \oplus P_j \\
P_j &= O_j \oplus C_j
\end{align*}
\]

### CTR

\[
\begin{align*}
O_j &= E_K(\text{CTR} + j) \\
C_j &= O_j \oplus P_j \\
P_j &= O_j \oplus C_j
\end{align*}
\]
CBC vs CFB

**CBC**
\[
C_0 = IV \\
C_j = E_K(C_{j-1} \oplus P_j)
\]
\[
P_1 = D_K(C_1) \oplus IV \\
P_j = D_K(C_j) \oplus C_{j-1}
\]

If specific bits of IV are flipped same bits of \( P_1 \) are flipped after decoding (Solution: Encrypt IV !)

If specific bits of \( C_2 \) are flipped same bits of \( P_2 \) are flipped, and random bits of \( P_3 \) are flipped after decoding

**CFB**
\[
C_0 = IV \\
C_j = E_K(C_{j-1}) \oplus P_j
\]
\[
P_1 = E_K(IV) \oplus C_1 \\
P_j = E_K(C_{j-1}) \oplus C_j
\]
Double DES

\[ C = E_{K_1}(E_{K_2}(P)) \]

Can we find a \( K_3 \) such that \( E_{K_1}(E_{K_2}(P)) = E_{K_3}(P) \)

- Highly improbable:
  - Each of the \( 2^{56} \) keys define a random mapping between two tables of size \( 2^{64} \)
  - \( 2^{64}! \) possible mappings – only a very small fraction (\( 2^{56} \)) is defined by single-DES
  - Repeated application of mappings is highly unlikely to produce a mapping provided by a single key!
Meet-in-the-Middle Attack

Unfortunately, double DES does not increase brute force complexity!

Meet-in-the-middle attack

\[ X = E_{K_1}(P) = D_{K_2}(C) \]

\( P_1 \) and \( C_1 \) known

Try \( X = D_K(C_1) \) for all \( 2^{56} \) possible values of \( K_2 \)

Sort the table (\( 2^{56} \) entries)

Compute \( E_K(P_1) \) for all \( 2^{56} \) possible values of \( K_1 \)

Values for which \( E_{K_1}(P_1) = D_{K_2}(C_1) \) are possible candidates

High probability of false alarm with only one known P-C pair

On an average \( 2^{112} \) \( / \) \( 2^{64} \) keys will map \( P_1 \) to the same \( C_1 \)

With two known P-C pairs probability of false alarm reduces to

\[ p_f = 2^{112} / 2^{64} / 2^{64} = 2^{-16} \]

With three known P-C pairs \( p_f = 2^{-80} \)
Triple DES

- Triple DES with three keys – brute-force complexity $2^{112}$

- Triple DES with two keys - equivalent to 80-bit security

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

*With $K_1 = K_2$, Triple DES becomes DES*

*Compatibility with old encrypters*

- Most commonly used
RC-4

- Stream Cipher.
- Extremely simple!
- Very fast – especially in software
- Easily adapts to any key length (1 byte to 256 bytes)
- Used in SSL / TLS
- WEP
- (Was) protected by trade secret – exposed (anonymously posted on the web) in 1994
Key Initialization

K[0] ....K[keylen-1] --- key bytes

For i = 0 to 255
  - S[i] = i;
  - T[i] = K[i mod keylen];

j = 0;

For i = 0 to 255
  - j = (j + S[i] + T[i]) mod 256;
  - SWAP(S[i], S[j]);

Throw away T, K; (retain S)
RC-4 – Stream Generation

- \( i, j = 0; \)
- while (true)
  - \( i = (i+1) \mod 256; \)
  - \( j = (j + S[i]) \mod 256; \)
  - SWAP(S[i],S[j]);
  - \( t = (S[i] + S[j]) \mod 256; \)
  - \( k = S[t]; \)
- The vector \( S \), at any time, is a random permutation of 1 to 256 (only swap performed on the vector).
void code(long* v, long* k) {
    unsigned long y=v[0],z=v[1], sum=0, /* set up */
    delta=0x9e3779b9, n=32 ;              /* a key schedule constant */
    while (n-->0) {                       /* basic cycle start */
        sum += delta ;
        y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
        z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;   /* end cycle */
    }
    v[0]=y ; v[1]=z ; }

void decode(long* v,long* k) {
    unsigned long n=32, sum, y=v[0], z=v[1],
    delta=0x9e3779b9 ;
    sum=delta<<5 ;  /* start cycle */
    while (n-->0)     {
        z-= (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
        y-= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
        sum-=delta ;  }
    /* end cycle */
    v[0]=y ; v[1]=z ;  }

Input v – 64 bits
As two 32 bit quantities
v[0], v[1]

k – 128 bits
As four 32 bit quantities
k[0],k[1],k[2],k[3]
Hash Functions

- $h = H(M)$
- $M$ can be of any size
- $h$ is always of fixed size
- Typically $h \ll \text{size}(M)$
- $h=H(x)$ is easy to compute given $x$
- Virtually impossible to calculate $x$ given $h$
- Weak collision resistance
  - Infeasible to find $x \neq y$ such that $H(x) = H(y)$
- Strong Collision resistance
  - Infeasible to find any $(x, y)$ such that $H(x) = H(y)$
Birthday Paradox

- 50 people in a room – what is the probability that two people have the same birthday?
- Extremely high – about 0.977
- A message $M$ hashes to $N$ bits – say $h$. What is the probability that another message $M_1$ hashes to $h$?
  - $1/2^N$ – we need to search $2^N$ to see a hit.
- What is the probability that two messages have the same hash?
  - We need to search only $2^{N/2}$ messages
- 64 bit hash is not strongly collision resistant
- Normally we use 128 or 160 bit hash functions
Compression Functions

- Hash functions are built using compression functions $C : \{0,1\}^{(m+t)} \rightarrow \{0,1\}^m$
- If the compression function satisfies the requirements (pre-image resistance and collision resistance) for some $t > 1$, we can construct a hash function using the Merkle-Damagard Construction
- Compression function used in MD5: $m=128$, $t=512$
MD5 – 128 bit hash
(Merkle-Damagard Construction)

- Message length $K$
- Pad message with $P$ bits such that $K+P$ is $448 \mod 512$ (64 bits less than a multiple of 512)
- Padding is done even if $K$ is already $448 \mod 512$
- Padding is 1 followed by $P-1$ zeros
- Length of padding is at least 1. Maximum value is 512
- Append length as a 64 bit value.
- Total length is $L \times 512$
- IV initialized to four fixed 32 bit quantities $A,B,C,D$
Each HMD5 block involves 64 rounds of data mangling
4 stages of 16 rounds each
Each stage has different compression functions F,G,H,I
Each round uses an entry from a fixed Table of length 64
Every bit of the hash code is a function of every bit of input

Other hash functions – SHA, SHA-1, RIPEMD-160
Keyed Message Authentication Code (HMAC)

- A shared key can be used to establish a private channel between the sender and the receiver
- It can also be used to authenticate messages
- In principle, HMAC for a message M, using a hash function h(), and a shared key K is computed as h(M,K).
- h(M,k) appended to the message
- Receiver (using the key K) can verify that
  - the sender has access to K
  - the message has not been modified en route
HMAC

- Special constructions based on hash functions are used for HMAC
- Standard HMAC using any hash function

\[
HMAC = h((K \oplus opad) || h((K \oplus ipad) || M))
\]

\[
\text{opad} = 0x5c5c \ldots 5c5c
\]

\[
\text{ipad} = 0x3636 \ldots 3636
\]

- CBC or OFB modes (with any block cipher) is also frequently used