Complementation Property of DES: Let E denote DES, and $\bar{x}$ the bitwise complement of x. Then $y = E_K(x)$ implies $\bar{y} = E_{\bar{K}}(\bar{x})$. That is, bitwise complementing both the key K and the plaintext x results in complemented DES ciphertext. Effect of complementing both x and k on

① **Key schedule:** Each key K_i is complemented.

② $(L_0, R_0) \leftarrow IP(m_1 \cdots m_{64})$: (L_0, R_0) are complemented.

③ **For each round of** $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$: Both R_{i-1} and K_i are complemented $\Rightarrow f(R_{i-1}, K_i)$ has no effect of complementation. L_i and R_i are complemented because:

$L_i = R_{i-1}$, and

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

④ $b_1 \cdots b_{64} \leftarrow (R_{16}, L_{16})$: R_{16} and L_{16} are complemented because of ③.

⑤ $C \leftarrow IP^{-1}(b_1 \cdots b_{64})$: C is complemented because of ③.

Key Recovery Attacks on Block Ciphers: We fix a block cipher $E: \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n$ having key-size k and block size n. It is assumed that the attacker knows the description of E and can compute it. A k-bit key T, called the target key, is chosen at random. Let $q \geq 0$ be some integer parameter.

Given: The adversary has a sequence of q input-output examples of E_T, say $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E_T(M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are all distinct n-bit strings.

Find: The adversary wants to find the target key T. We say that a key K is consistent with the input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$ if $E_K(M_i) = C_i$ for all $1 \leq i \leq q$. Let $\text{Cons}_E((M_1, C_1), \ldots, (M_q, C_q))$ be the set of all keys consistent with the input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$. The adversary wants to find a key $T \in \text{Cons}_E((M_1, C_1), \ldots, (M_q, C_q))$.

The two common types of attacks are:

(1) **Known Plaintext Attack**: $M_1, \ldots, M_q$ are any distinct points; the adversary has no control over them, and must work with whatever it gets.

(2) **Chosen Plaintext Attack**: $M_1, \ldots, M_q$ are chosen by the adversary. We assume that the adversary has access to an oracle for the function E_K. It can feed the oracle M_1 and get back $C_1 = E_K(M_1)$. It can then decide on a value M_2, feed the oracle this, and get back C_2, and so on.

Exhaustive Key Search: The adversary goes through all possible keys $K' \in \{0,1\}^k$ until it finds one that explains the input-output pairs. For $i = 1, \ldots, 2^k$ let T_i denote the i-th k-bit string (in lexicographic order).

$E_K S_E((M_1, C_1), \ldots, (M_q, C_q))$

{ for $i = 1, \ldots, 2^k$ do

{ if($E(T_i, M_1) = C_1$ AND $E(T_i, M_2) = C_2$ - - - AND $E(T_i, M_q) = C_q$) then

{ return T_i }

}

}

Worst case complexity of exhaustive key search is 2^k.

Average case complexity is:

$$\sum_{i=1}^{2^k} i \cdot \Pr[k = T_i] = \sum_{i=1}^{2^k} \frac{i}{2^k} = \frac{1}{2^k} \sum_{i=1}^{2^k} i = \frac{1}{2^k} \cdot \frac{2^k(2^k+1)}{2}$$

$$= \frac{2^k+1}{2} \approx 2^{k-1}.$$

Thus to make key-recovery by exhaustive search computationally prohibitive, one must make the key length k of the block cipher large enough.

Security of DES: Assuming that there is a VLSI chip that can compute DES at the rate of 1.6 G bits/sec. Since a DES plaintext is 64 bits, the chip enables us to perform $(1.6 \times 10^9)/64 = 2.5 \times 10^7$ DES computations per second. To perform 2^{55} computations ($k = 56$), we need $2^{55}/(2.5 \times 10^7) \approx 1.44 \times 10^9$ sec $= 45.7$ years

complementation property of DES can be exploited to reduce the search time by half to 22.8 years. Parallel computers can break DES easily. In 1993, Weiner showed that one can design a $1 million machine that does the exhaustive key search for DES in about 3.5 hours on average. For improving the security of DES against exhaustive key search, we should increase the key size. There are three methods to increase the key size of DES:

① Double-DES: Let k_1, k_2 be 56-bit DES keys and let M be a 64-bit plaintext. Let

$2DES(k_1 \| k_2, M) = DES(k_2, DES(k_1, M))$.

This defines a block cipher $2DES: \{0,1\}^{112} \times \{0,1\}^{64} \to \{0,1\}^{64}$ that we call Double-DES. It has a 112-bit key, viewed as consisting of two 56-bit DES keys.

2DES is invertible:

$2DES^{-1}(k_1 \| k_2, C) = DES^{-1}(k_1, DES^{-1}(k_2, C))$

The key length of 112 seems to be large enough for exhaustive key search. But effective key length of 2DES is only 57 bits due to Meet in the Middle Attack: The attacker, given M_1, C_1 is attempting to find $k_1 \| k_2$. We observe that

$C_1 = DES(k_2, DES(k_1, M_1)) \Rightarrow DES^{-1}(k_2, C_1) = DES(k_1, M_1)$.

This leads to the following attack. For $i = 1, \ldots, 2^{56}$ we let T_i denote the i-th 56-bit string (in lexicographic order).

```
M in M_2DES(M_1, C_1)
{ for i = 1, ..., 2^56 do
  { L(i) ← DES(T_i, M_1) }
  for j = 1, ..., 2^56 do
  { R(j) ← DES^-1(T_j, C_1) }
  S ← { (i,j) : L(i) = R(j) }
  Pick some (l,r) ∈ S and return T_l || T_r
}
```

For any $(i,j) \in S$ we have:
$DES(T_i, M_1) = L(i) = R(j) = DES^{-1}(T_j, C_1)$
and as a consequence $DES(T_j, DES(T_i, M_1)) = C_1$.
So the key $T_i || T_j$ is consistent with the input-output example (M_1, C_1). Thus

$$\{T_l || T_r : (l,r) \in S\} = Cons_E((M_1, C_1)).$$

The attack picks some pair (l,r) from S and outputs $T_l || T_r$, thus returning a key consistent with the input-output example (M_1, C_1). The attack makes $2^{56} + 2^{56} = 2^{57}$ DES or DES^{-1} computations. The step of forming the set S can be implemented in linear time in the size of the arrays involved by using hashing. Thus the running time is dominated by the 2^{57} DES and DES^{-1} computations making effective key length of 2DES to be 57 bits.

② Triple-DES : The triple-DES ciphers use three iterations of DES or DES^{-1}. The three-key variant is defined by:

$3DES_3(k_1 \| k_2 \| k_3, M) = DES(k_3, DES^{-1}(k_2, DES(k_1, M)))$,

so that $3DES_3 : \{0,1\}^{168} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{64}$.

The two-key variant is defined by:

$3DES_2(k_1 \| k_2, M) = DES(k_2, DES^{-1}(k_1, DES(k_2, M)))$,

so that $3DES_2 : \{0,1\}^{112} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{64}$.

The middle operation is DES^{-1} to make triple DES a generalization of DES:

$DES(k, M) = 3DES_3(k \| k \| k, M)$,

$DES(k, M) = 3DES_2(k \| k, M)$.

Due to meet in the middle attack, $3DES_3$ has effective bit length of $\log_2(2^{56} + 2^{112}) = 112$ bits.

Meet in the middle attack is not effective in $3DES_2$ because its effective bit length of $\log_2(2^{56} + 2^{112}) = 112$ bits is same as the actual bit length of 112 bits.

(3) DESX : 2DES, 3DES3, and 3DES2 seem to provide more security by increasing the effective key length. But they have one disadvantage that they are slow as compared to DES. The first is twice as slow as DES and the other two are three times as slow. DESX is having same speed as DES, but it provides more security against meet in the middle attack because its effective key length is 120 bits.

Let k_1, k_2 be 64-bit strings, and let M be a 64-bit plaintext. Let:

$DESX(k \| k_1 \| k_2, M) = k_2 \oplus DES(k, k_1 \oplus M)$.

This defines a block cipher $DESX: \{0,1\}^{184} \times \{0,1\}^{64} \to \{0,1\}^{64}$.

It has a 184-bit key, viewed as consisting of a 56-bit DES key plus two auxiliary keys, each 64 bits long. DESX is invertible:

$DESX^{-1}(k \| k_1 \| k_2, C) = k_1 \oplus DES^{-1}(k, k_2 \oplus C)$.

Due to meet in the middle attack, DESX has effective key length of $\log_2(2^{64} + 2^{120}) = 120$ bits.