DES Encryption Modes

- The encryption mode that was assumed in the DES analysis discussed so far, is the Electronic-Codebook or ECB, which operates on 64-bit blocks of data at a time.

- In many systems however, there is a requirement to deal with much larger blocks of data or with streams of data, which makes the default 64-bit structure very cumbersome.

- To expand the functionality of DES, other modes of encryption were designed into the algorithm, such as:

    - Cipher-Block Chaining (CBC) (Block ciphers)
    - Cipher-Feedback Mode (CFM) (Stream Ciphers)
    - Output Feedback (OFB) (Stream Ciphers)
    - Counter Mode (CTR) (Stream Ciphers)

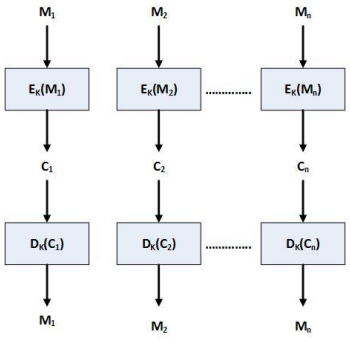
ECB Mode

- The messages are broken up into blocks of 84 bits, and processed independently.

- Each block is encrypted/decrypted separately:

  Encryption: $C_i = E_K(M_i)$
  Decryption: $M_i = D_K(C_i)$

- The following diagram illustrates the ECB mode of DES:



DES - ECB Mode of Operation

- As covered earlier, this mode of operation is deterministic; the same data block gets encrypted the same way, which means that patterns of data will be revealed when a data block repeats.

- It is also malleable, which means that reordering ciphertext results in reordered plaintext.

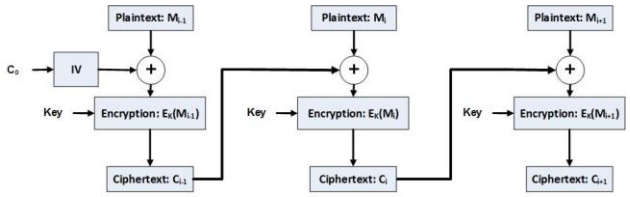
- The advantage here is that errors in one ciphertext block will not propagate through the rounds.

CBC Mode

- In CBC mode, the **next** input depends upon **previous** output; described mathematically as:

  Encryption: $C_i = E_K(M_i \oplus C_{i-1});\ C_0 = IV$
  Decryption: $M_i = C_{i-1} \oplus D_K(C_i);\ C_0 = IV$

- The encryption process is illustrated diagrammatically as follows:



CBC Encryption Process

- This mode of operation produces randomized encryption; repeated plaintext gets mapped to different ciphertext.

- A ciphertext block depends on all preceding plaintext blocks; which means that reordering affects the decryption process.

- Errors in one block propagate to two blocks; one bit error in C_j affects all bits in M_j and one bit in M_{j+1}

- This mode carries out sequential encryption, which means that parallelization is very difficult to implement.

- This mode can be proven to be "secure" assuming that **random** IV's are used. Random IVs protect the integrity of IV and ciphertext. Consider the following observation (assuming no IV):

  - If, $C_i = C_j\ ;\ then, E_K(M_i \oplus C_{i-1}) = E_K(M_i \oplus C_{j-1})$
  - And so, $(M_i \oplus C_{i-1}) = (M_i \oplus C_{j-1})$
  - Thus, $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$

Using DES to construct Stream Ciphers

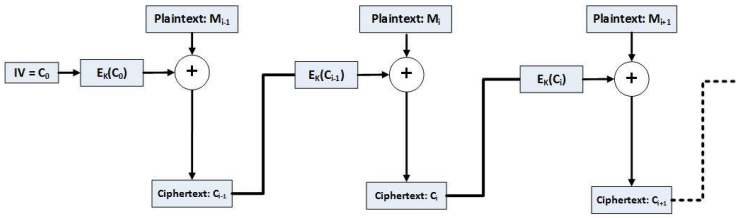
- DES can also be used to construct stream ciphers such as:
    - Cipher Feedback (CFB)
    - Output Feedback (OFB)
    - Counter Mode (CTR)
- There are some common properties of these ciphers that must be kept in mind:
- In the mode DES uses only the encryption function $E_K(M)$ of the cipher both for encryption and decryption
- The process in malleable, meaning it is possible to make predictable bit changes by feeding it known plaintext.

Cipher Feedback (CFB)

- In CFB mode, the message is XOR'd with the feedback of encrypting the previous block:

  Encryption: $C_i = E_K(C_{i-1}) + M_i;\ C_0 = IV$

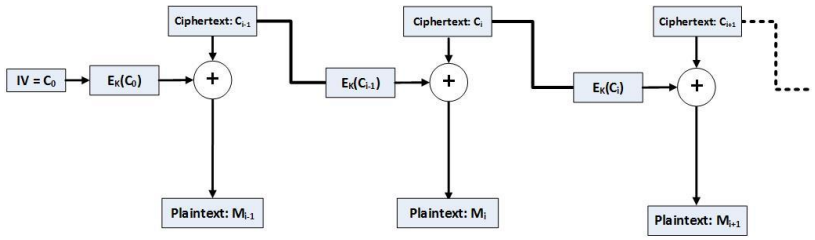
- This can be illustrated as follows:



DES - CFB Encryption Process

- The decryption process is described mathematically and diagrammatically as follows:

  Decryption: $M_i = C_i \oplus E_K(C_{i-1});\ C_0 = IV$

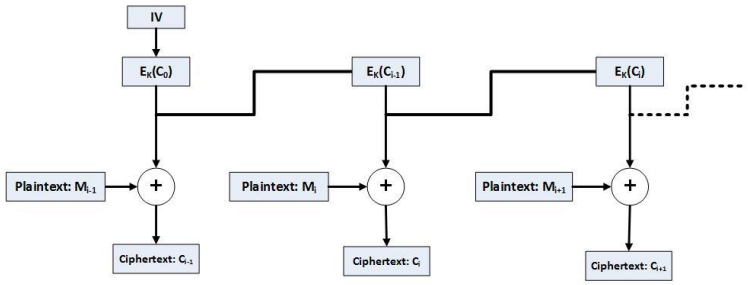


DES - CFB Decryption Process

- Note that the same encryption function E_k is used here also for the decryption process.
- The main advantage of this technique is that it provides randomized encryption, which is very difficult to cryptanalyze.
- In addition, each ciphertext block depends on all preceding plaintext blocks; thus reorder affects decryption, which again make it difficult to cryptanalyze.
- The main disadvantage is that errors propagate for several blocks after the error, but the mode is self-synchronizing (like CBC).
- This technique is slow, which means a decreased throughput.
- The encryption process is sequential and therefore slow. It is very difficult to parallelize the operations.

Output Feedback (OFB)

- OFB constructs a Pseudo Random Number Generator using DES encryption ($E_K(M)$) function. The following diagram illustrates this technique:



DES - OFB Encryption Process

- The main advantage of this technique is that it provides randomized encryption, which is very difficult to cryptanalyze.
- The technique is sequential encryption just like CFB, but pre-processing possible in this model, which will speed the process up.
- Error propagation limited using this mode.
- Subject to limitations of stream ciphers, which are more difficult to implement correctly, and prone to weaknesses based on usage - since the principles are similar to one-time pad, the keystream has very strict requirements.

Counter Mode (CTR)

- CTR mode is yet another way to construct a Pseudo Random Number Generator (PRNG) using DES.
- Mathematically it is described as follows:

  Encryption: $C_i = M_i \oplus E_K(nonce + i)$
  Decryption: $M_i = C_i \oplus E_K(nonce + i)$

- A nonce is equivalent to a random number, similar to an IV.
- Sender and receiver both share a **nonce** (does not need to be secret), and the secret key **K**.
- CTR mode has following advantages:
  - **Software and hardware efficiency**: different blocks can be encrypted in parallel.
  - **Preprocessing:** the encryption part can be done offline and when the message is known, just do the XOR.
  - **Random access**: decryption of a block can be done in random order, very useful for disk encryption.
  - **Messages of arbitrary length**: ciphertext is the same length with the plaintext (i.e., no IV).

Securing Block Ciphers

- Use variable key lengths.
- Use mixed operators rather than XOR only; use more than one arithmetic and/or Boolean; this can provide non-linearity.
- Use data dependent rotation, i.e., round vary depending on the data.
- Key-dependent S-boxes.
- Lengthy key schedule algorithm.
- Variable plaintext/ciphertext block length
- Variable number of rounds
- Operation on both data halves each round
- Variable *f()* function (varies from round to round)
- Key-dependent rotation

DES Security

- Due to the way that the initial key is modified to get a subkey for each round of the algorithm, certain initial keys are **weak keys**.

- Remember that the initial value is split into two halves, and each half is shifted independently. If all the bits in each half are either 0 or 1, then the key used for any cycle of the algorithm is the same for all the cycles of the algorithm.

- This can occur if the key is entirely 1s, entirely 0s, or if one half of the key is entirely 1s and the other half is entirely 0s.

- In addition, two of the weak keys have other properties that make them less secure (see text references).

- The four weak keys are shown below in hexadecimal notation (remember that every eighth bit is a parity bit):

  01010101 01010101
  FEFEFEFE FEFEFEFE
  E0E0E0E0 F1F1F1F1
  1F1F1F1F 0E0E0E0E

- The outcome of using weak keys is that the output the Permuted Choice 1 (PC1) in the DES key schedule leads to the round keys (K_1….K_{16}) being either **all zeros, all ones** or **alternating zero-one** patterns.

- Additionally, some pairs of keys encrypt plaintext to the identical ciphertext. In other words, one key in the pair can decrypt messages encrypted with the other key in the pair.

- This is due to the way in which DES generates subkeys; instead of generating 16 different subkeys, these keys generate only two different subkeys. Each of these subkeys is used eight times in the algorithm.

- Since all the subkeys are identical, and DES is a Feistel network, the encryption function becomes self-inverting; that is, encrypting twice with a weak key K produces the original plaintext:

  $E_K(E_K(x)) = x$ for all x, i.e., the encryption and the decryption are the same.

- These keys are called **semiweak keys,** and are shown in hexadecimal notation below:

  01FE 01FE 01FE 01FE and FE01 FE01 FE01 FE01
  1FE0 1FE0 0EF1 0EF1 and E01F E01F F10E F10E
  01E0 01E0 01F1 01F1 and E001 E001 F101 F101
  1FFE 1FFE 0EFE 0EFE and FE1F FE1F FE0E FE0E
  011F 011F 010E 010E and 1F01 1F01 0E01 0E01
  E0FE E0FE F1FE F1FE and FEE0 FEE0 FEF1 FEF1

- However, putting things into perspective, weak and semi-weak keys are not considered "fatal flaws" of DES. There are 2^{56} (7.21×10^{16}) possible keys for DES, of which only four are weak and twelve are semi-weak.

Brute Force Attacks

- Known-Plaintext Attack (the cryptanalyst knows one or several pairs of ciphertext and the corresponding plaintext.)

- This attack attempts all 2^{56} possible keys to find a match.

- To demonstrate this attack, RSA Security devised a series of DES challenges to be attempted using a series of brute force attack contests using a message format: "**The secret message is: xxxxxxxx**"

- The first challenge was in 1997 (thousands of volunteers connected by Internet); it was solved in 96 days (3 months). The message was "**The secret message is: Many hands make light work**."

- In 1998 EFF (Electronic Frontier Foundation, a non-profit organization) machine (costs $250K): 3 days.

- 1999 (distributed.net and Deep Crack, combined): 22 hours and 15 minutes. The message was "**See you in Rome (second AES Conference, March 22-23, 1999**".

Dictionary Attacks

- Each plaintext may result in 2^{64} different ciphertexts, but there are only 2^{56} possible different key values.

- Start by encrypting the known plaintext with all possible keys.

- Maintain a look-up table of size 2^{56}.

- Given a Plaintext/Ciphertext pair (M,C), look up C in the table