Data Encryption Standard (DES)

- The most widely used private key block cipher, is the Data Encryption Standard (DES).
- It was adopted in 1977 by the National Bureau of Standards now (NIST) as Federal Information Processing Standard 46 (FIPS PUB 46).
- DES encrypts data in 64-bit blocks using a 56-bit key.
- The DES enjoys widespread use.
- It has also been the subject of much controversy its security.

DES history

- In the late 1960s, IBM set up a research project in computer cryptography led by Horst Feistel.
- The project concluded in 1971 with the development of the LUCIFER algorithm.
- LUCIFER is a Feistel block cipher that operates on blocks of 64 bits, using a key size of 128 bits.
- Because of the promising results produced by the LUCIFER project, IBM embarked on an effort, headed by Walter Tuchman and Carl Meyer, to develop a marketable commercial encryption product that ideally could be implemented on a single chip.

- It involved not only IBM researchers but also outside consultants and technical advice from NSA.
- The outcome of this effort was a refined version of LUCIFER that was more resistant to cryptanalysis but that had a reduced key size of 56 bits, to fit on a single chip.
- In 1973, the National Bureau of Standards (NBS) issued a request for proposals for a national cipher standard.
- IBM submitted the modified LUCIFER.
- It was by far the best algorithm proposed and was adopted in 1977 as the Data Encryption Standard.
**DES Design Controversy**

- although DES standard is public, and before its adoption as a standard, DES faced considerable controversy (arguing) over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished especially in financial applications
- Recent analysis has shown despite this controversy, that DES is well designed.
- DES is theoretically broken using Differential or Linear Cryptanalysis but in practise is unlikely to be a problem yet.

**DES overall structure**

- DES takes 64-bits of data as input and as a key.
- process for enciphering a 64-bit data block which consists of:
  1. an initial permutation (IP) which shuffles the 64-bit input block
  2. 16 rounds of a complex key dependent round function involving substitutions & permutations
  3. a final permutation, being the inverse of IP

- Also rapid advances in computing speed though have rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman.
- It is still standardized for legacy systems, with either AES or triple DES for new applications.
• handling of the 56-bit key and consists of:
  1. an initial permutation of the key (PC1) which selects 56-bits out of the 64-bits input, in two 28-bit halves
  2. 16 stages to generate the 48-bit subkeys using a left circular shift and a permutation of the two 28-bit halves

1. Initial Permutation IP
   • The initial permutation and its inverse are defined by tables, in next slide
   • The input to a table consists of 64 bits numbered left to right from 1 to 64.
   • The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
   • Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

(a) Initial Permutation (IP)

<table>
<thead>
<tr>
<th>58</th>
<th>50</th>
<th>42</th>
<th>34</th>
<th>26</th>
<th>18</th>
<th>10</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>52</td>
<td>44</td>
<td>36</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>62</td>
<td>54</td>
<td>46</td>
<td>38</td>
<td>30</td>
<td>22</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>64</td>
<td>56</td>
<td>48</td>
<td>40</td>
<td>32</td>
<td>24</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>57</td>
<td>49</td>
<td>41</td>
<td>33</td>
<td>25</td>
<td>17</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>59</td>
<td>51</td>
<td>43</td>
<td>35</td>
<td>27</td>
<td>19</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>61</td>
<td>53</td>
<td>45</td>
<td>37</td>
<td>29</td>
<td>21</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>63</td>
<td>55</td>
<td>47</td>
<td>39</td>
<td>31</td>
<td>23</td>
<td>15</td>
<td>7</td>
</tr>
</tbody>
</table>

(b) Inverse Initial Permutation (IP⁻¹)

<table>
<thead>
<tr>
<th>40</th>
<th>8</th>
<th>48</th>
<th>16</th>
<th>56</th>
<th>24</th>
<th>64</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>7</td>
<td>47</td>
<td>15</td>
<td>55</td>
<td>23</td>
<td>63</td>
<td>31</td>
</tr>
<tr>
<td>38</td>
<td>6</td>
<td>46</td>
<td>14</td>
<td>54</td>
<td>22</td>
<td>62</td>
<td>30</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>45</td>
<td>13</td>
<td>53</td>
<td>21</td>
<td>61</td>
<td>29</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>44</td>
<td>12</td>
<td>52</td>
<td>20</td>
<td>60</td>
<td>28</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>43</td>
<td>11</td>
<td>51</td>
<td>19</td>
<td>59</td>
<td>27</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>42</td>
<td>10</td>
<td>50</td>
<td>18</td>
<td>58</td>
<td>26</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>41</td>
<td>9</td>
<td>49</td>
<td>17</td>
<td>57</td>
<td>25</td>
</tr>
</tbody>
</table>
• Note that the bit numbering for DES reflects IBM mainframe practice, and is the opposite of what we now mostly use - so be careful!
• Numbers from Bit 1 (leftmost, most significant) to bit 32/48/64 etc (rightmost, least significant).
• Note that examples are specified using hexadecimal.
• Here a 64-bit plaintext value of “675a6967 5e5a6b5a” (written in left & right halves) after permuting with IP becomes “ffb2194d 004df6fb”.
• IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

- function F, takes R half & subkey, and processes them through E, add subkey, S & P.
- This follows the classic structure for a feistel cipher.
- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:
  \[ L_i = R_{i-1} \]
  \[ R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \]

- F takes 32-bit R half and 48-bit subkey:
  - expands R to 48-bits using perm E
  - adds to subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P
- Note that the s-boxes provide the “confusion” of data and key values,
• The R input is first expanded to 48 bits by using expansion table E that defines a permutation plus an expansion that involves duplication of 16 of the R bits.
• The resulting 48 bits are XORed with Ki.
• This 48-bit result passes through a substitution function comprising 8 S-boxes which each map 6 input bits to 4 output bits, producing a 32-bit output, which is then permuted by permutation P.

**Substitution Boxes S**
- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (row bits) select one row of 4
  - inner bits 2-5 (col bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)
- example:
  - \[ S(18 \ 09 \ 12 \ 3d \ 11 \ 17 \ 38 \ 39) = 5fd25e03 \]
For example, in $S_1$, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

The example lists 8 6-bit values (ie 18 in hex is 011000 in binary, 09 hex is 001001 binary, 12 hex is 010010 binary, 3d hex is 111101 binary etc), each of which is replaced following the process detailed above using the appropriate S-box. ie

- $S_1$(011000) lookup row 00 col 1100 in $S_1$ to get 5
- $S_2$(001001) lookup row 01 col 0100 in $S_2$ to get 15 = $f$ in hex
- $S_3$(010010) lookup row 00 col 1001 in $S_3$ to get 13 = $d$ in hex
- $S_4$(111101) lookup row 11 col 1110 in $S_4$ to get 2 etc
**DES Key Schedule**

- The DES Key Schedule generates the subkeys needed for each data encryption round.
- The 64-bit key input is first processed by **Permuted Choice One**.
- The resulting 56-bit key is then treated as two 28-bit quantities C & D.
- In each round, these are separately processed through a **circular left shift** (rotation) of 1 or 2 bits.

- These shifted values serve as input to the next round of the key schedule.
- They also serve as input to **Permuted Choice Two** which produces a 48-bit output that serves as input to the round function F.
- The 56-bit key size comes from security considerations as we know now.
- It was big enough so that an exhaustive key search was about as hard as the best direct attack.

- The extra 8 bits were then used as parity (error detecting) bits, which makes sense given the original design use for hardware communications links.
- However, we hit an incompatibility with simple software implementations since the top bit in each byte is 0 (since ASCII only uses 7 bits), but the DES key schedule throws away the bottom bit! A good implementation needs to be cleverer!
DES Decryption

- As with any Feistel cipher, DES decryption uses the same algorithm as encryption except that the subkeys are used in reverse order SK16 .. SK1.
- If you trace through the DES overview diagram you can see how each decryption step top to bottom with reversed subkeys, undoes the equivalent encryption step moving from bottom to top.

Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.
- If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.
- DES exhibits a strong avalanche effect.
Strength of DES – Key Size

- With a key length of 56 bits, there are $2^{56}$ possible keys, which is approximately $7.2 \times 10^{16}$ keys.
- Thus a brute-force attack appeared impractical.
- In July 1998, the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000.
- The attack took less than three days.
- The EFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98].

- There have been other demonstrated breaks of the DES using both large networks of computers & dedicated h/w, including:
  - 1997 on a large network of computers in a few months
  - 1998 on dedicated h/w (EFF) in a few days
  - 1999 above combined in 22hrs!
- It is important to note that there is more to a key-search attack than simply running through all possible keys.
- Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext.
- Clearly must now consider alternatives to DES, the most important of which are AES and triple DES.
Strength of DES – Analytic Attacks

• Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.
• The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration.
• These techniques utilise some deep structure of the cipher by gathering information about encryptions so that eventually you can recover some/all of the sub-key bits, and then exhaustively search for the rest if necessary.
• Generally these are statistical attacks which depend on the amount of information gathered for their likelihood of success.
• Attacks of this form include differential cryptanalysis, linear cryptanalysis, and related key attacks.

Strength of DES – Timing Attacks

• timing attacks relate to public-key algorithms.
• However, the issue may also be relevant for symmetric ciphers.
• A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.
• A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
• though DES appears to be fairly resistant to a successful timing attack.

Differential Cryptanalysis

• Biham & Shamir show Differential Cryptanalysis can be successfully used to cryptanalyse the DES with an effort on the order of $2^{47}$ encryptions, requiring $2^{47}$ chosen plaintexts.
• They also demonstrated this form of attack on a variety of encryption algorithms and hash functions.
• Differential cryptanalysis was known to the IBM DES design team as early as 1974 (as a T attack), and influenced the design of the S-boxes and the permutation P to improve its resistance to it.
• Compare DES's security with the cryptanalysis of an eight-round LUCIFER algorithm which requires only 256 chosen plaintexts, verses an attack on an eight-round version of DES requires $2^{14}$ chosen plaintexts.

• The differential cryptanalysis attack is complex.
• The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block.
• Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round, which means you cannot trace values back through cipher without knowing the value of the key.
• Differential Cryptanalysis compares two related pairs of encryptions, which can leak information about the key, given a sufficiently large number of suitable pairs.
• with a known difference in the input
• searching for a known difference in output
• when same subkeys are used

\[
\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1} \\
= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\
= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)]
\]

• The AES analysis process has highlighted this attack approach, and showed that it is a concern particularly with smartcard implementations,

Linear Cryptanalysis

• A more recent development is linear cryptanalysis.
• This attack is based on finding linear approximations to describe the transformations performed in DES.
• This method can find a DES key given $2^{43}$ known plaintexts, as compared to $2^{47}$ chosen plaintexts for differential cryptanalysis.
• Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES.

• The objective of linear cryptanalysis is to find an effective linear equation relating some plaintext, ciphertext and key bits that holds with probability $p < 0.5$ as shown.

\[
P[i_1, i_2, \ldots, i_b] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c]
\]

where $i_a, j_b, k_c$ are bit locations in $P, C, K$

• Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the equation for a large number of plaintext-ciphertext pairs, in order to determine whether the sum of the key bits is 0 or 1, thus giving 1 bit of info about them.
• This is repeated for other equations and many pairs to derive some of the key bit values.
• Because we are dealing with linear equations, the problem can be approached one round of the cipher at a time, with the results combined.

What is good in DES Design
• 8 S-boxes provide for
  — non-linearity
  — resistance to differential cryptanalysis
  — good confusion
• 3 permutation P provide for
  — increased diffusion
• number of rounds
  — more is better, exhaustive search best attack
• function f:
  — provides “confusion”, is nonlinear, avalanche
  — have issues of how S-boxes are selected
• key schedule
  — complex subkey creation, key avalanche

Feistel cipher design aspects
• The cryptographic strength of a Feistel cipher derives from three aspects of the design:
  — the number of rounds,
  — the function F, and
  — the key schedule algorithm.
• number of rounds,
  • The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
  • In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack.
  • This criterion is attractive because it makes it easy to judge the strength of an algorithm and to compare different algorithms.
• the function F,
  • The function F provides the element of confusion in a Feistel cipher, want it to be difficult to “unscramble” the substitution performed by F.
  • One obvious criterion is that F be nonlinear.
  • The more nonlinear F, the more difficult any type of cryptanalysis will be.
  • We would like it to have good avalanche properties, or even the strict avalanche criterion (SAC).
  • Another criterion is the bit independence criterion (BIC).
  • One of the most intense areas of research in the field of symmetric block ciphers is that of S-box design.
  • Would like any change to the input vector to an S-box to result in random-looking changes to the output.
  • The relationship should be nonlinear and difficult to approximate with linear functions.
The key schedule algorithm.
With any Feistel block cipher, the key schedule is used to generate a subkey for each round.
Would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
The key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

### Data Encryption Standard summary

Dr. Ahmed M. ElShafee

---

<table>
<thead>
<tr>
<th>(a) Initial Permutation (IP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>58  50  42  34  26  18  10  2</td>
</tr>
<tr>
<td>60  52  44  36  28  20  12  4</td>
</tr>
<tr>
<td>62  54  46  38  30  22  14  6</td>
</tr>
<tr>
<td>64  56  48  40  32  24  16  8</td>
</tr>
<tr>
<td>57  49  41  33  25  17  9</td>
</tr>
<tr>
<td>59  51  43  35  27  19  11  3</td>
</tr>
<tr>
<td>61  53  45  37  29  21  13  5</td>
</tr>
<tr>
<td>63  55  47  39  31  23  15  7</td>
</tr>
</tbody>
</table>
### (b) Inverse Initial Permutation (IP⁻¹)

<table>
<thead>
<tr>
<th>40</th>
<th>8</th>
<th>48</th>
<th>16</th>
<th>56</th>
<th>24</th>
<th>64</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>7</td>
<td>47</td>
<td>15</td>
<td>55</td>
<td>23</td>
<td>63</td>
<td>31</td>
</tr>
<tr>
<td>38</td>
<td>6</td>
<td>46</td>
<td>14</td>
<td>54</td>
<td>22</td>
<td>62</td>
<td>30</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>45</td>
<td>13</td>
<td>53</td>
<td>21</td>
<td>61</td>
<td>29</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>44</td>
<td>12</td>
<td>52</td>
<td>20</td>
<td>60</td>
<td>28</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>43</td>
<td>11</td>
<td>51</td>
<td>19</td>
<td>59</td>
<td>27</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>42</td>
<td>10</td>
<td>50</td>
<td>18</td>
<td>58</td>
<td>26</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>41</td>
<td>9</td>
<td>49</td>
<td>17</td>
<td>57</td>
<td>25</td>
</tr>
</tbody>
</table>

### (c) Expansion Permutation (E)

<table>
<thead>
<tr>
<th>32</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>
### Definition of DES S-Boxes

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
<th>$S_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 4 13 1 2 15 11 8 10 6 12 5 9 0 7</td>
<td>15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10</td>
<td>12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11</td>
<td>7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15</td>
<td>13 7 9 3 4 6 10 2 8 5 14 12 11 15 1</td>
<td>4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8</td>
<td>3 13 4 7 15 2 3 8 14 12 0 1 10 6 9 11 5</td>
<td>10 15 4 2 7 12 9 5 6 1 13 0 14 3 0 11</td>
<td>13 0 11 7 4 9 1 10 14 3 5 12 3 15 11 8</td>
<td>1 14 11 13 12 3 7 14 10 15 6 8 0 5 9 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1 18 14 10 13 5 8 7 2 6 12 5 9 0 15</td>
<td>15 8 10 1 3 15 4 2 11 6 7 12 0 5 10 9</td>
<td>11 15 4 2 7 12 9 5 6 1 13 0 14 3 0 11</td>
<td>7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8</td>
<td>2 1 14 7 4 10 8 13 15 12 9 0 3 5 2 12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12</td>
<td>5 2 8 14</td>
<td>6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12</td>
<td>13 2 8 4 10 13 7 11 15 13 1 3 4 15</td>
<td>2 1 14 7 4 10 8 13 15 12 9 0 3 5 2 12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Permutation Function (P)

<table>
<thead>
<tr>
<th>16 7 20 21 29 12 28 17</th>
<th>1 2 3 4 5 6 7 8</th>
<th>9 10 11 12 13 14 15 16</th>
<th>17 18 19 20 21 22 23 24</th>
<th>25 26 27 28 29 30 31 32</th>
<th>33 34 35 36 37 38 39 40</th>
<th>41 42 43 44 45 46 47 48</th>
<th>49 50 51 52 53 54 55 56</th>
<th>57 58 59 60 61 62 63 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 15 23 26 5 18 31 10</td>
<td>2 8 24 14 32 27 3 9</td>
<td>2 14 7 4 10 8 13 15 12 9 0 3 5 2 12</td>
<td>16 7 20 21 29 12 28 17</td>
<td>9 10 11 12 13 14 15 16</td>
<td>17 18 19 20 21 22 23 24</td>
<td>25 26 27 28 29 30 31 32</td>
<td>33 34 35 36 37 38 39 40</td>
<td>41 42 43 44 45 46 47 48</td>
</tr>
</tbody>
</table>
### (b) Permed Choice One (PC-1)

<table>
<thead>
<tr>
<th>57</th>
<th>49</th>
<th>41</th>
<th>33</th>
<th>25</th>
<th>17</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58</td>
<td>50</td>
<td>42</td>
<td>34</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>59</td>
<td>51</td>
<td>43</td>
<td>35</td>
<td>27</td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>3</td>
<td>60</td>
<td>52</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>63</td>
<td>55</td>
<td>47</td>
<td>39</td>
<td>31</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>62</td>
<td>54</td>
<td>46</td>
<td>38</td>
<td>30</td>
<td>22</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>61</td>
<td>53</td>
<td>45</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>13</td>
<td>5</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>

### (c) Permed Choice Two (PC-2)

<table>
<thead>
<tr>
<th>14</th>
<th>17</th>
<th>11</th>
<th>24</th>
<th>1</th>
<th>5</th>
<th>3</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>6</td>
<td>21</td>
<td>10</td>
<td>23</td>
<td>19</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>8</td>
<td>16</td>
<td>7</td>
<td>27</td>
<td>20</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>41</td>
<td>52</td>
<td>31</td>
<td>37</td>
<td>47</td>
<td>55</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>51</td>
<td>45</td>
<td>33</td>
<td>48</td>
<td>44</td>
<td>49</td>
<td>39</td>
<td>56</td>
</tr>
<tr>
<td>34</td>
<td>53</td>
<td>46</td>
<td>42</td>
<td>50</td>
<td>36</td>
<td>29</td>
<td>32</td>
</tr>
</tbody>
</table>

### (d) Schedule of Left Shifts

<table>
<thead>
<tr>
<th>Round Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits Rotated</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**DES Cipher Example**

Simplified 2 rounds operated on 26 characters English plaintext characters space

Dr. Ahmed M. ElShafee
Example 1

Encrypt the following message using simplified 2 rounds DES cipher operated on 26 English plaintext chars space "supplies" using key "secret key".

- PC1 = beagdfc
- PC2 = befcda
- IP = bheagdfc
- EP =dacba 
- CP = dacb 
- P = dacb
Key generation

PC1: beagdfc
PC2: befcdad

Left shift
PC1: beagdfc
PC2: befcdad

Left shift
K1
K2

Key generation

PC1: beagdfc
PC2: befcdad

Left shift
PC1: beagdfc
PC2: befcdad

Left shift
K1
K2

Encryption Round 1

IP: bheagdfc
CP: bdca
P: dacb

EP: dacb
R1

EP: dacb
R1

Encryption Round 1

IP: bheagdfc
CP: bdca
P: dacb

EP: dacb
R1

R2

R2
Example 3

- Decrypt the following message using simplified 2 rounds DES cipher operated on 26 English plaintext chars pace “qkhnbyt”, using key “secretky”
  - PC1 = beagdfc
  - PC2 = befcdad
  - IP = bheagdfc
  - EP = dacbad
  - CP = bdca
  - P = dacb
Decryption Round 2

Encryption Round 1

Key generation

Empty Forms
Five Modes of Operation

- Electronic codebook mode (ECB)
- Cipher block chaining mode (CBC) *
- Cipher feedback mode (CFB)
- Output feedback mode (OFB)
- Counter mode (CTR)

When ciphers opened the file in a image program they where able to get a rough outline of the image.
This happens due to problems in how repetitive data is encrypted and is a problem with all major encryption standards.

Electronic Code Book (ECB)

- When you encrypt 2 identical blocks of data you can expect it to have the same output.
- Images can have large blocks of color that are identical.
- Because of this when rendered in an image program you can get an outline like above.
- Because of this there was a need for new ways to encrypt data so that it appears as completely unreadable data
- This called mode of operations

The plaintext is broken into blocks, $P_1, P_2, P_3, ...$
- Each block is encrypted independently:
  $$C_i = E_k(P_i)$$

Electronic Codebook (ECB) mode encryption
• For a given key, this mode behaves like we have a gigantic codebook, in which each plaintext block has an entry, hence the name Electronic Code Book

• **Strength:** it’s simple.

• **Weakness:**
  - Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.
  - If the same message (e.g., an SSN) is encrypted (with the same key) and sent twice, their ciphertexts are the same.
  - Typical application: secure transmission of short pieces of information (e.g. a temporary encryption key)

---

**Cipher Block Chaining (CBC)**

• The plaintext is broken into blocks: $P_1, P_2, P_3, \ldots$

• Each plaintext block is XORed (chained) with the previous ciphertext block before encryption (hence the name):

\[ C_i = E_K(C_{i-1} \oplus P_i) \]

\[ C_0 = IV \]

• Use an Initial Vector (IV) to start the process.

• Decryption: $P_i = C_{i-1} \oplus D_K(C_i)$

• Application: general block-oriented transmission.

---

• The encryption of a block depends on the current and all blocks before it.

• So, repeated plaintext blocks are encrypted differently.

• **Initialization Vector (IV)**
  - Must be known to both the sender & receiver
  - Typically, IV is either a fixed value or is sent encrypted in ECB mode before the rest of ciphertext.
Cipher Feedback (CFB) Mode

- The block cipher is used as a stream cipher.
- Appropriate when data arrives in bits/bytes.
- \( s \) can be any value; a common value is \( s = 8 \).
- A ciphertext segment depends on the current and all preceding plaintext segments.
- A corrupted ciphertext segment during transmission will affect the current and next several plaintext segments.
  - How many plaintext segments will be affected?

Output feedback mode (OFB)

- The block cipher is used as a stream cipher.
- Appropriate when data arrives in bits/bytes.
- \( s \) can be any value; a common value is \( s = 8 \).
- A ciphertext segment depends on the current and all preceding plaintext segments.
• The block cipher is used as a stream cipher.
• Appropriate when data arrives in bits/bytes.
• Advantage:
  – more resistant to transmission errors; a bit error in a ciphertext segment affects only the decryption of that segment.
• Disadvantage:
  – Cannot recover from lost ciphertext segments; if a ciphertext segment is lost, all following segments will be decrypted incorrectly (if the receiver is not aware of the segment loss).
• IV should be generated randomly each time and sent with the ciphertext.
• **Strengths:**
  – Needs only the encryption algorithm
  – Fast encryption/decryption; blocks can be processed (encrypted or decrypted) in parallel; good for high speed links
  – Random access to encrypted data blocks
• IV should not be reused.