CLASS NOTES

E-COMM & E-SECURITY

TEACHING PHASE -I

**1) Substitution Ciphers**

**1.1 Ceaser cipher:**

Plain text: “this is ddu”

**Algorithm** 🡪 **requires key**

Cipher text: “bfdug sjefhjsg ”

26

Key : 3

**Encryption:**

Plain text: ABC

Algo: plain text + key

Cipher : DEF

Decryption:

Cipher: DEF

Algo : cipher text – key

Plain text: ABC

Prof. Tanvi Goswami, DD University, Nadiad

**1.2 Monoalphabetic cipher:**

- Keyword

- Mapping

Keyword: INDIA

PLAIN TEXT MSG: HELLO

A B C D E F G H . . . Z I N D A B C E F…………………………..

Cipher text: FBXXY

EX:

KEY WORD: EZRA CORNELL

PLAIN TEXT: HELLO

CIPHER TEXT?

| A | B | C | D | E | F. | G. | H | …. | Z |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | Z | R | A | C | O | N | L | B | . |

- Map H with L

- Map E with C

And so on

Cipher Text is: LC….

Prof. Tanvi Goswami, DD University, Nadiad

**1.3 MONOALPHABETIC CIPHER USING COLUMNAR TRANSPOSITION**

Keyword: CORNELL 🡪

1 2 3 4 5 6

C O R N E L

A B D F G H

I J K M P Q

S T U V W X

Y Z

KEY WORD: CORNELL

PLAIN TEXT: HELLO

A B C D E F G H . . . . Z C A I S Y O B J T Z - Map H with J

- Map E with Y

And so on

Cipher text: JY…

HELLO 🡪 JY

Prof. Tanvi Goswami, DD University, Nadiad

Ex:

KEYWORD: ZEBRAS

PLAIN TEXT: WE ARE DISCOVERED CIPHER TEXT: ??

Prof. Tanvi Goswami, DD University, Nadiad

**1.4 Affine cipher :**

It is a monoalphabetic substitution cipher

Consider following algorithm for Encryption:

1) Transform each of the letters in the plaintext alphabet to the corresponding integer in the range 0 to m-1. Consider this integer as “x”.

2) With this done, the encryption process for each letter is given by: E(x) = (ax+b) mod m.

Note: where a and b are the key for the cipher and m is number of alphabets. Using the using the key a= 5, b = 8 transform the plain text “**JULY**” into cipher text.

**Sol: ??**

**CIPHER TEXT:**

****

**The encryption function for this example will be *y* = *E*(*x*) = (5*x* + 8) mod 26**

Prof. Tanvi Goswami, DD University, Nadiad

 Prof. Tanvi Goswami, DD University, Nadiad

**1.5 PLAYFAIR CIPHER**

**- Polygraphic substitution cipher**

**BALLOON**

**BA Lx LO ON Nx**

| **M** | **O** | **N** | **A** | **R** |
| --- | --- | --- | --- | --- |
| **C** | **H** | **Y** | **B** | **D** |
| **E** | **F** | **G** | **i/j** | **K** |
| **L** | **P** | **Q** | **S** | **T** |
| **U** | **V** | **W** | **X** | **Z** |

**NA DI AD**

**ENCRYPTED MSG: AR BK RB**

**DECRYPT: NA DI AD**

Ex:

Keyword: SUCCESS

1) KEYWORD MATRIX

2) ENCRYPT 🡪 WE ARE DISCOVERED SAVE YOURSELF 3) DECRYPT 🡪 QTQBAUBOTICN

ANS:

Prof. Tanvi Goswami, DD University, Nadiad

| **S** | **U** | **C** | **E** | **A** |
| --- | --- | --- | --- | --- |
| **B** | **D** | **F** | **G** | **H** |
| **I/J** | **K** | **L** | **M** | **N** |
| **O** | **P** | **Q** | **R** | **T** |
| **V** | **W** | **X** | **Y** | **Z** |

MSG: WE ARE DISCOVERED SAVE YOURSELF

1) WE AR ED IS CO VE RE DS AV EY OU RS EL FX YU ET UG OB

**ENCRYPTION:**

2) YU ET UG OB SQ YS YG BU SZ GE PS OE MC LC

**DECRYPTION:**

3) PR OF ES SI ON AL

QT QB AU BO TI CN

Example 2:

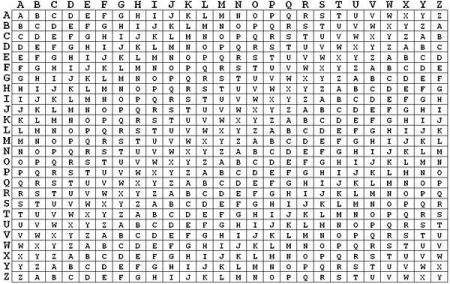
Keyword: information

Plain text: computer security

Prof. Tanvi Goswami, DD University, Nadiad

**1.6 VIGENERE CIPHER**

**VIGENERE TABLE**

****

🡺 To encrypt, pick a letter in the plaintext and its corresponding letter in the keyword,

🡺 Use the keyword letter and the plaintext letter as the row index and column index, respectively, and the entry at the row-column intersection is the letter in the ciphertext.

EX:

MSG: DHARMSINH DESAI UNIVERSITY

KEYWORD: NADIAD

MAPPING:

Prof. Tanvi Goswami, DD University, Nadiad

P: D H A R M S I N H D E S A I U N I V E R S I T Y K: N A D I A D N A D I A D N A D I A D N A D I A D C: Q H D Z . . . . . . . . . . .

To encrypt: use the keyword letter and the plaintext letter as the row index and column index

EX:

MSG: HELLO

KEYWORD: HI

H E L L O 🡪

H I H I H

Prof. Tanvi Goswami, DD University, Nadiad

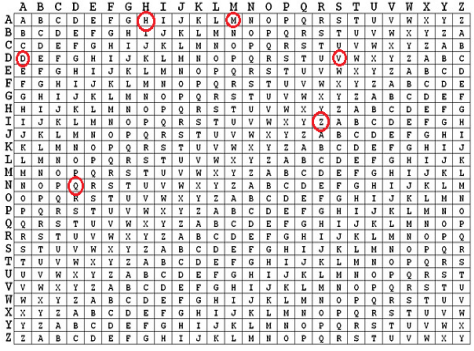
**1.7 ONE TIME PAD:**

key length = msg length; no repetition

2) random keys 🡪 not related to plain text

3) space is also considered

4) diff to generate random keys

Adadgdfhjkoadgef -> Such pattern in key is not allowed Prof. Tanvi Goswami, DD University, Nadiad

**1.8 HILL CIPHER**

- Polygraphic substitution cipher

- Based on linear algebra

- Key matrix - matrix of numbers

- 3x3 matrix 🡪 3 letters at time

- 5x5 matrix 🡪 5 letters at time

- Matrix inverse

**Encrypt: M \* P = (result) mod 26** 🡪 **cipher text**

**Decrypt: Inverse of key matrix \* C = (result) mod 26** 🡪 **plain text**

| **2** | **4** | **5** |
| --- | --- | --- |
| **9** | **2** | **1** |
| **3** | **17** | **7** |

| **0** |
| --- |
| **19** |
| **19** |

| **171** |
| --- |
| **57** |
| **456** |

**X = Mod26**

**171 Mod 26** 🡪**15** 🡪**P**

**57 Mod 26—> 5** 🡪 **F**

**456 Mod 26**🡪 **14** 🡪 **O**

**Plain text: ATT ACK ATD AWN**

**3X3** 🡪 **3 LETTERS AT TIME FOR ENCRYPTION Hello** 🡪 **hel lox** 🡪 **use fillers**

**Hell** 🡪 **hel lxx**

**1) ATT ACK ATD AWN**

**2) A**🡪 **0, B-> 1, C**🡪 **2 …………………Z**🡪**25**

Prof. Tanvi Goswami, DD University, Nadiad

**3) A T T A C K A T D A W N P F O**

Decryption using HILL cipher:

Suppose the key matrix is:

| 1 | 2 | 3 |
| --- | --- | --- |
| 0 | 1 | 4 |
| 5 | 6 | 0 |

CT: DZYNAG

Find the plain text PT

PT = (Inverse of key matrix \* CT ) mod 26

Prof. Tanvi Goswami, DD University, Nadiad

**2) TRANSPOSITION TECHNIQUE**

**2.1 RAILFENCE CIPHER**

MSG: HELLO HI NADIAD

DEPTH: 2

|  | H | L | O | I | A | I | D |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | E | L | H | N | D | A |  |  |

CIPHER TEXT:

🡺READ ROW WISE:

🡺CIPHER TEXT: HLOIAID ELHNDA

ENCRYPTION: ARRANGE DIAGONALLY, READ ROW WISE

DECRYPTION: ARRANGE ROW WISE , READ

DIAGONALLY

**EXAMPLE**:

PLAIN TEXT: INFORMATION SECURITY

DEPTH: 3

CIPHER TEXT: ??

🡺More complex way is right a plain text in a rectangle, row by row and read it column by column

Ex: keyword: APPLE

Assign no. according to its order

A P P L E

1 4 5 3 2

Prof. Tanvi Goswami, DD University, Nadiad

Suppose msg length: 28

Key length: 5

Then number of rows in rectangle matrix is 28/5 = 6

This might give columns of diff length

Plain text : this is a columnar transposition

KEY: 14532

1 4 5 3 2

| T | H | I | S | I |
| --- | --- | --- | --- | --- |
| S | A | C | O | L |
| U | M | N | A | R |
| T | R | A | N | S |
| P | O | S | I | T |
| I | O | N | x | x |

TO encrypt: read columns according to key number

1st: TSUTPI ILRST SOANI HAMROO ICNASN

Input for 2nd permutation

1 4 5 3 2

| T | S | T | P | I |
| --- | --- | --- | --- | --- |
| I | L | R | S | T |
| S | O | A | N | I |
| H | A | M | R | O |
| O | I | C | N | A |
| S | N |  |  |  |

READ COL WISE:

Prof. Tanvi Goswami, DD University, Nadiad

**Cipher: TISHOS ITIOA PSNRN SLOANI TRAMC** EX:

KEYWORD: MATH

PT: CODES ARE COOL

DOUBLE COLUMNAR TRANSPOSITION

EX: DECRYPT THE FOLLOWING CT

KEYWORD: LAYER

EOMNT SCKUE ECWOL IYRXT TOERW

Using DOUBLE COLUMNAR TRANSPOSITION

Prof. Tanvi Goswami, DD University, Nadiad

**DATA ENCRYPTION STANDARD (DES)**

🡺DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text.

🡺 The same algorithm and key are used for encryption and decryption, with minor differences. The key length is **56 bits**. The basic idea is show in figure. **2^56 attempt**

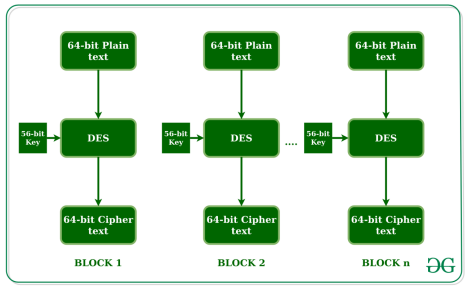
🡺DES is based on the two fundamental attributes of cryptography: **substitution** (also called as confusion) and **transposition** (also called as diffusion).

🡺DES consists of **16 steps**, each of which is called as a **round**. Each round performs the steps of substitution and transposition.

PLAIN TEXT : BLOCK SIZE: 64 BITS

KEY LEGTH: 56 BITS

Prof. Tanvi Goswami, DD University, Nadiad



1. In the first step, the 64 bit plain text block is handed over to an initial Permutation (IP) function.

1010 🡪 4,3,1,2 🡪 4TH BIT is promoted to 1st position

3rd bit is promoted to 2nd position

1st bit is demoted to 3rd position

2nd bit is demoted to 4th position

🡺0 1 1 0

2. The initial permutation performed on plain text.

3. Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).

32🡪 LPT, 32🡪 RPT

4. Now each LPT and RPT to go through 16 rounds of encryption process.

Prof. Tanvi Goswami, DD University, Nadiad

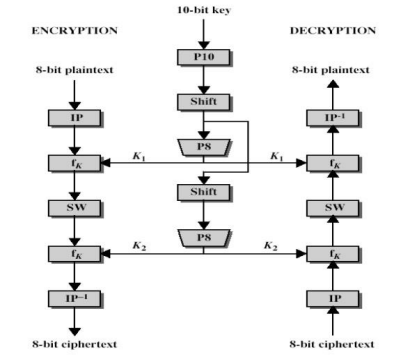
5. In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block

6. The result of this process produces 64 bit cipher text.

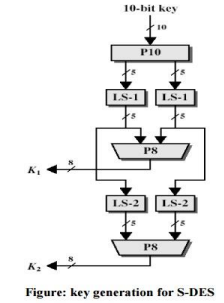
**SIMPLIFIED DES / S-DES**

**Std DES**🡪 **64 bit block, 56 bit key**

**s-des**🡪 **8 bit block, 10 bit key**

**** Prof. Tanvi Goswami, DD University, Nadiad

1) S-des KEY generation algorithm



K1, k2 🡪 subkeys (8 bit) 🡪 obtained from the main key 10bit Ex:

**1.1 P-10 permutation**

10 bit key is: 10100 00010

1 2 3 4 5 6 7 8 9 10

| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

P10 permutation sequence given is:

Bit number

3 5 2 7 4 10 1 9 8 6

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Prof. Tanvi Goswami, DD University, Nadiad

**1.2 Left shift operation on two halves independently**

LS-1 🡪 left shift 1

1 2 3 4 5 6 7 8 9 10

| 1 | 0 | 0 | 0 | 0 |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 |

**1.3 apply P8 permutation**

Ls-1

| 0 | 1 | 1 | 0 | 0 |
| --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 |

Suppose P-8 sequence given is: 6,3,7,4,8,5,10,9

| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- |

KEY (SUBKEY: 1) OR K1 : 1010 0100

For subkey 2: perform 2-bit position left shift operation on each string obtained from step number **1.2**

**LS-2** 🡪 **LEFT SHIFT 2**

| 0 | 0 | 0 | 0 | 1 |
| --- | --- | --- | --- | --- |

Ls-1

| 1 | 1 | 0 | 0 | 0 |
| --- | --- | --- | --- | --- |

1 2 3 4 5 6 7 8 9 10

| 0 | 0 | 1 | 0 | 0 |
| --- | --- | --- | --- | --- |

Ls-2

| 0 | 0 | 0 | 1 | 1 |
| --- | --- | --- | --- | --- |

**Then apply P-8 permutation on the output to obtain subkey:2**

Prof. Tanvi Goswami, DD University, Nadiad

P-8 sequence given is: 6,3,7,4,8,5,10,9

6 3 7 4 8 5 10 9

| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| --- | --- | --- | --- | --- | --- | --- | --- |

**KEY (SUBKEY:2) OR K2: 0100 0011**

2) S-DES ALGORITHM:

STEP 1) Initial permutation (IP)

STEP 2) apply complex function Fk1 (using 1st subkey) on left 4 bits

STEP 3) Swap(SW) left and right 4 bits of text (left 4 bits becomes right 4 bits and right 4 bits will be on left side , now modify left 4 bits)

STEP 4) apply complex function Fk2 (using 2nd key) on left 4 bits STEP 5) apply inverse of IP (IP-1)

STEP:1 : IP (INITIAL PERMUTATION)

🡺Input to this 1st step is 8 bit plain text.

🡺Suppose plain text is: 0100 1100

🡺And IP sequence given is: 2,6,3,1,4,8,5,7

Prof. Tanvi Goswami, DD University, Nadiad

| Bit  positions | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plain text | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| P-8 Permutation | | | | | | | | |
| P-8  sequence | 2 | 6 | 3 | 1 | 4 | 8 | 5 | 7 |
| Permuted text | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

1st step op ( IP): 1100 0010

IP INVERSE (IP-1)

SUPPOSE IP IS

| BIT  POSITIONS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP | 2 | 6 | 3 | 1 | 4 | 8 | 5 | 7 |

- IP-1 Simply puts the bits back where they come from - The 1st bit position is occupied by 2nd bit, reverse is 2nd bit position occupied by 1st bit

| BIT  POSITIONS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP-1 |  | 1 |  |  |  | 2 |  |  |

- 2nd bit position occupied by 6th bit🡪 reverse 6th bit position occupied by 2nd bit

| BIT  POSITIONS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP-1 |  | 1 |  |  |  | 2 |  |  |

.

Prof. Tanvi Goswami, DD University, Nadiad

Final IP-1sequence would be

| BIT  POSITIONS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP-1 | 4 | 1 | 3 | 5 | 7 | 2 | 8 | 6 |

**STEP:2 : Fk (the complex function)**

- Most complex function

- Substitution + transposition

- Can be written as:

**Fk (L,R) = L + F(R,SK1), R**

Sk1 = subkey 1, L: Left bits, R: Right bits, + is xor

For ex:

Output of IP stage is :

PT is: 0100 1100

Output of IP stage is: 1100 0010

K1: 0 0 0 1 0 1 1 1

- Modify half of the bits at a same time

- 1100 🡪 L or left 4 bits

- 0010 --> R or right 4 bits

**Fk (L,R) = L + F(R,SK1), R**

Fk1 (1100, 0010) = 1100 + F(0010, 0 0 0 1 0 1 1 1) , 0010 L R k1 R

E/P : Expansion permutation 🡪 k1 8 bit , R- 4 bits

R 🡪 0010 🡪

4 1 2 3 2 3 4 1 🡪 E/P sequence

Prof. Tanvi Goswami, DD University, Nadiad

0 0 0 1 0 1 0 0

🡺Apply xor operation of K1 with R (8 bits)

K1: 0 0 0 1 0 1 1 1

e/p: 0 0 0 1 0 1 0 0

0 0 0 0 0 0 1 1

1ST ROW : 0000

2ND ROW: 0011

🡺Divide this 8 bit o/p in 2 rows

1st row: 0 0 0 0 🡪 1st , 4th bit 🡪 00 🡪 0 🡪 row number 2nd, 3rd bit 🡪 00 🡪 0 🡪 col number

🡺0th row and 0th column

2nd row: 0 0 1 1 🡪 1st, 4th bit 🡪 01 🡪 1 🡪 row num 2nd, 3rd bit 🡪 01 🡪 1 🡪 col num

🡺SUBSTITUTION

S-boxes 🡪 substitution boxes

S0

0 1 2 3

| 1 | 2 | 3 | 2 |
| --- | --- | --- | --- |
| 2 | 3 | 1 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 3 | 2 |

OP FROM S0 = 1 = 01

Prof. Tanvi Goswami, DD University, Nadiad

S1 s-box

0 1 2 3

| 1 | 0 | 3 | 2 |
| --- | --- | --- | --- |
| 3 | 2 | 1 | 0 |
| 0 | 2 | 1 | 3 |
| 3 | 1 | 3 | 2 |

0P FROM S1 = 2 = 10

S0S1 = 0110 => OUTPUT FROM COMPLEX FUNCTION Fk1

S0 S1 L

0110 + 1100

Apply P4 (SOS1 XOR L)= 2,4,3,1

**Fk (L,R) = L + F(R,SK1), R**

Fk1 (1100, 0010) = 1100 + F(0010, 0 0 0 1 0 1 1 1) , 0010 L R k1 R

Fk1 (1100, 0010) = 0110, 0010

Prof. Tanvi Goswami, DD University, Nadiad

**3) SWAP**

0010, 0110

**4) Fk (L,R) = L + F(R,SK2), R**

**Fk2 (0010, 0110) = 0010 + F(0110, 0100 0011) , 0110**

**E/P(0110) =** 4 1 2 3 2 3 4 1

**0 0 1 1 1 1 0 0**

**K2 = 0100 0011**

**EP = 0011 1100**

**= 0111 1111**

**SO=> 0111 => 01 => 1ST ROW => 1 => 01 11 => 3RD COL**

**S1 => 1111 => 11 => 3RD ROW 3RD COL => 10 SOS1 => 0110**

P4 (SOS1)= 2,4,3,1

= 1 0 1 0

L + 1010 => **0010 + 1010 => 1000**

FINAL OP FROM FK2 = 1000 0110

Prof. Tanvi Goswami, DD University, Nadiad

**5) IP-1**

IP = 1000 0110

| BIT  POSITIONS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP-1 | 4 | 1 | 3 | 5 | 7 | 2 | 8 | 6 |
|  | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

**CIPHER TEXT: 0100 1001**

**S-DES ENCRYPTION SAMPLE**

| To the input (plaintext), apply initial permutation IP:  IP  2 6 3 1 4 8 5 7  In the next steps, we will develop 4 bits with which to replace the left half of this "blue" result. | Input:  0 1 1 0 1 1 0 1  1 1 1 0 0 1 1 0 |
| --- | --- |

Prof. Tanvi Goswami, DD University, Nadiad

To right 4 bits of above result, apply expansion/permutation E/P

(generating 8 bits from 4). The bit numbering is that of the 4-bit right nibble, not of the 8-bit byte (e.g., indicated bit 2 refers to byte's bit 6).

E/P

0 0 1 1 1 1 0 0

| 4 | 1 |
| --- | --- |

| 2 | 3 |
| --- | --- |

| 2 |
| --- |

| 3 | 4 |
| --- | --- |

1

| Upon above result, perform binary XOR operation with  subkey K1:  K1  1 0 1 0 0 1 0 0 |
| --- |

| Determine a row and a column from above XOR result. For the row, combine bits 1 and 4 and convert to decimal. For the column, combine bits 2 and 3 and convert to decimal.  Determine another row and column. For this second row, combine bits 5 and 8; for this second column, bits 6 and 7.  Identify the entry in s-box S0 at the first row/first column you determined. S0 shows it in decimal; convert it to binary (two bits). Enter those bits as the first half of the 4-bit number at right. Identify the entry in s-box S1 at |
| --- |

1 0 0 1 1 0 0 0

left nibble:

bits 1 & 4 -> 11 -> 3

bits 2 & 3 -> 00 -> 0

therefore, get from S0 row 3 col 0 result is 3 -> 11

right nibble:

bits 1 & 4 -> 10 -> 2

bits 2 & 3 -> 00 -> 0

therefore, get from S1 row 2 col 0 result is 3 -> 11

1 1 1 1

Prof. Tanvi Goswami, DD University, Nadiad

the second row/second column you determined. Convert it to binary; enter those two bits as the second half of the number at right.

|  |
| --- |

| c0 | c1 | c2 | c3 |
| --- | --- | --- | --- |

S0

| r0 | 1 | 0 | 3 | 2 |
| --- | --- | --- | --- | --- |

=r1 3 2 1 0 r2 0 2 1 3 r3 3 1 3 2

|  |
| --- |

| c0 | c1 | c2 | c3 |
| --- | --- | --- | --- |

S1 =

r0 0 1 2 3 r1 2 0 1 3 r2 3 0 1 0 r3 2 1 0 3

| To above result, apply permutation P4:  P4  2 4 3 1 |
| --- |

1 1 1 1

Prof. Tanvi Goswami, DD University, Nadiad

| Upon the above P4 result, perform binary XOR operation, combining it with the left 4-bits of our first result (application of IP to original plaintext input, blue cell above).  We are trying to replace the left half of that first result. These XOR result bits are the replacement bits for it. |
| --- |

| Rewrite that "blue" first result with its left half replaced. (Look it up, keep/copy its right half, use the preceding result as the new left half.) |
| --- |

| Swap the two 4-bit halves of the above (previous) result.  In the next steps, we will again develop 4 replacement bits, and with them replace the left half of this "green" swap result. The steps will be the same ones used for that purpose already. |
| --- |

| To right 4 bits of above swap result, apply expansion/permutation E/P (generating 8 bits from 4):  E/P  4 1 2 3 2 3 4 1 |
| --- |

XOR with 1110

0 0 0 1

0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1

1 0 0 0 0 0 1 0

Prof. Tanvi Goswami, DD University, Nadiad

Upon above result, perform binary XOR operation with

subkey K2:

| K2  0 1 0 0 0 0 1 1 |
| --- |

Determine a row and a column from above result. For the row, combine bits 1 and 4 and convert to decimal. For the column, combine bits 2 and 3 and convert to decimal.

Determine another row and column. For this second row, combine bits 5 and 8; for this second column, bits 6 and 7.

Identify the entry in s-box S0 at the first row/first column you determined. It's given in decimal; convert it to binary (two bits). Enter those bits as the first half of the 4-bit number at right. Identify the entry in s-box S1 at the second row/second column you determined. Convert it to binary; enter those two bits as the second half of the number at right.

| S0  = | c0 c1 c2 c3 | | | | |  |
| --- | --- | --- | --- | --- | --- | --- |
| r0 1 0 3 2 | | | | |  |
| r1 3 2 1 0 | | | | |  |
| r2 | 0 | 2 | 1 | 3 |  |

1 1 0 0 0 0 0 1

left nibble:

bits 1 & 4 -> 10 -> 2

bits 2 & 3 -> 10 -> 2

therefore, get from S0 row 2 col 2 result is 1 -> 01

right nibble:

bits 1 & 4 -> 01 -> 1

bits 2 & 3 -> 00 -> 0

therefore, get from S1 row 1 col 0 result is 2 -> 10

0 1 1 0

Prof. Tanvi Goswami, DD University, Nadiad

S1 =

r3 3 1 3 2

|  | c0 | c1 | c2 | c3 |
| --- | --- | --- | --- | --- |

r0 0 1 2 3

| r1 | 2 | 0 | 1 | 3 |
| --- | --- | --- | --- | --- |

r2 3 0 1 0 r3 2 1 0 3

| To above result, apply permutation P4:  P4  2 4 3 1 |
| --- |

| Upon the above P4 result, perform binary XOR operation, combining it with the left 4-bits of the earlier swap result (green cell above).  We are trying to replace the left half of that swap result. These XOR result bits are the replacement bits for it. |
| --- |

| Rewrite that "green" swap result with its left half replaced. (Look it up, keep/copy its right half, use the preceding result as the new left half.) |
| --- |

1 0 1 0

XOR with 0110

1 1 0 0

1 1 0 0 0 0 0 1

Prof. Tanvi Goswami, DD University, Nadiad

| To above result, apply reverse of initial permutation IP, which is IP-1:  IP-1  4 1 3 5 7 2 8 6  This result is ciphertext. It is the S DES encryption of the plaintext input. | 0 1 0 0 0 1 1 0 |
| --- | --- |

Prof. Tanvi Goswami, DD University, Nadiad

**BLOCK CIPHER MODES OF OPERATIONS**

1) Electronic code book (ECB)

2) Cipher block chaining (CBC)

3) Cipher Feedback (CFB)

4) Output Feedback (OFB)

5) Counter (CTR)

🡺Basic building algo. For providing data security 🡪 DES Data: 11110000 🡪 2^8 (00000000 to 11111111)

Key: 1010101010 🡪 2^10 (00000 00000 to 11111 11111) Data: 0000 0000 🡪 000000 000000 🡪 c1

00000 00001 🡪 c2

64 bit block 🡪 2^64

56 bit key 🡪 2^56

Data: 0000

Input block: 0001 -> 2 bit key

Key : 00,01,10,11

| 0001 | 00 | X |
| --- | --- | --- |
|  | 01 | X1 |
|  | 10 | X2 |
|  | 11 | X3 |
| 0010 | 00 | Y |
|  | 01 | Y1 |
|  | 10 | Y2 |
|  | 11 | Y3 |
|  |  |  |

Prof. Tanvi Goswami, DD University, Nadiad

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| 1111 | 00 |  |
|  | 01 |  |
|  | 10 |  |
|  | 11 | Z3 |

0010

1111

Block length : b

S no. of bits

Keep: s bits

Discard : b-s bits

1 byte of data:

Keep: 8 bits

Discard: 64-8 bits

Prof. Tanvi Goswami, DD University, Nadiad

Double DES - encryption

K1 K2

PT CT

E **E**

Decryption:

K2 K1

CT PT

D **D**

Triple DES

🡺With 2 keys

1) ENCRYPTION

K1 K2 K1

PT CT E **D** E

1) DECRYPTION

K1 K2 K1

PT CT D **E** D

🡺TRIPLE DES 🡪 With 3 keys

K1 K2 K3

Prof. Tanvi Goswami, DD University, Nadiad

PT CT E **D** E

MEET IN MIDDLE ATTACk

🡪 Targets block cipher cryptographic fuc.

C= Ek2(Ek1(p))

X = Ek1(p) == Dk2( c)

K1 K2

PT x CT

E **E**

x

pair (p,c)

1) Encrypt P for all 2^56 possible values of k1

2) Stores all the result in a table & then sort the table by the values of x

3) Decrypt C using all 2^56 possible values of k2

4) Check the result against the table for match

5) K1-k2, (p2,c2) 🡪

64 bit 🡪 2^64 🡪

56+56 = 112 bit keys

🡺True alarms

🡺False alarms 🡺 2^112 / 2^64 🡪 2^48

1st try 🡪 2^48 false alarms

2nd try🡪 2^48 / 2^64 🡪 2^-16 --. False alarm rate reduced by 2^- 16 times

.

Prof. Tanvi Goswami, DD University, Nadiad

.

.

. correct pair of k1 and k2

Q: A certain application uses s- DES algorithm in CBC mode in order to secure the data. If the secure key is given as 10101 11001 and Initialization vector (IV) is 1010 0101 then find out the cipher text for the plain text 1001 0110 0011 0010.

Consider following information:

P10: 5, 7, 9, 10, 8, 3, 1, 2, 4, 6

P8: 9, 7, 1, 8, 2, 3, 4, 6

IP: 2, 6, 4, 1, 8, 3, 7, 5

E/P: 4, 1, 2, 3, 1, 2, 4, 3

P4: 4 ,3 ,2 , 1

|  |  |  | c0 c1 c2 c3 |
| --- | --- | --- | --- |

|  |
| --- |

|  | c0 | c1 | c2 | c3 |
| --- | --- | --- | --- | --- |
|  | | | | |
| r0 | 0 | 1 | 2 | 3 |
|  | | | | |
| r1 | 2 | 0 | 1 | 3 |
|  | | | | |
| r2 | 3 | 0 | 1 | 0 |
|  | | | | |
| r3 | 2 | 1 | 0 | 3 |

S0 =

| r0 | 1 | 0 | 3 | 2 |
| --- | --- | --- | --- | --- |
|  | | | | |
| r1 | 3 | 2 | 1 | 0 |
|  | | | | |
| r2 | 0 | 2 | 1 | 3 |
|  | | | | |
| r3 | 3 | 1 | 3 | 2 |

S1 =

Prof. Tanvi Goswami, DD University, Nadiad

ADVANCED ENCRYPTION STANDARD (AES)

1) Pre- round transformation

Data 🡪 block (128bit) 🡪 round key is added Based on key length num of rounds differs

1.1 sub byte substitution

1.2 shift rows

1.3 mix columns

1.4 Add round key

🡺PT 🡪 in a 4\*4 matrix 🡪 1 Byte

.

.

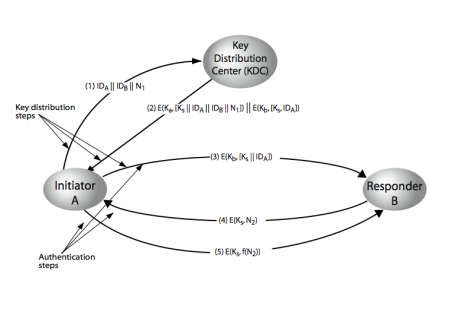
.

..

Key length : 128 bit 🡪 10, 192 bit 🡪 12, 256bit 🡪 14

Prof. Tanvi Goswami, DD University, Nadiad

**CENTRALIZED KEY DISTRIBUTION SCENARIO USING KDC (Key distribution center)**

****1) A🡪 KDC : identifier of A & B. unique identifier/ nonce N1 / timestamp / random num (true / psudo)

2) KDC 🡪A: one time session key Ks which is encrypted using master key of A Ka

+ copy of request (IDa, IDb, N1) + (session key Ks and IDa encrypted using master key of B which is Kb.)- only decrypted by B .

3) A🡪B: forward the msg as it is to B 🡪 Ekb(Ks, IDa)

Prof. Tanvi Goswami, DD University, Nadiad

4) B🡪A: As a sign of ACK B will send a nonce N2 which is encrypted using session key Ks which was delivered during step 3.

5) A🡪B: respond with f(N2) 🡪 mathematical func will be applied on nonce N2 🡪 encrypted using Ks

After it is received at B 🡪 B will again apply the func to retrieve N2 🡪 the N2 sent and N2 received are same 🡪 data is not modified and comm. Is secure

**DECENTRALIZED KEY DISTRIBUTION**

1.

**Initiator A Responder**

2.

**B**

3.

1) A🡪 B : Request | N1

2) B 🡪A : Ekm [Ks || Request II IDB || f(N1) || N2 ]

🡺Session key

🡺Ack : copy of req

🡺IDb

🡺F(N1) 🡪 modified N1

🡺New nonce N2

🡺All above information is encrypted using Master key shared between A & B. Km / kab

3) A🡪 B: Ks [ f(N2) ] : send Ack to B 🡪 after receiving session key Ks

Prof. Tanvi Goswami, DD University, Nadiad

N number of entities they are connected with then they must have master keys as many as users with which it is connected 🡪 n(n-1)/2 Kac

Kad

Kae

RANDOM NUMBER GENERATORS

RAMDOM

NUMBER

GENERATION

True Random no. Generator

(TRNG)

Pseudo Random no. Generator

(PRNG)

Pseudo Random Function

(PRF)

Linear

Congruential

Algorithm

Characteristics of PRNG: 1) Randomness:

Blum Blum

Shub Generator

🡺Bit stream generated must be highly random even though it is deterministic.

🡺Uniformity: 1’s and 0’s must be equal

Prof. Tanvi Goswami, DD University, Nadiad

🡺Scalability: sequence as well as subseq. Must be random 🡺Consistency: behavior of generator must be consistent with any value of SEED (starting value)

2) Unpredictability:

Forward : 1110001111010 \_\_

& Backward: seed 🡪 PRNG algo 🡪 Random num

Seed must be secure

By knowing the Random no we cannot obtain/predict the seed value 3) Char. Of seed

1) PRNG

*Xn*+1 = (*aX*n + *c*) mod *m*

M : modulus ; m>0

A: multiplier; 0<a<m

C: increment ; 0<=c<m

X0: starting value/ seed

X1: (aX0 +c) mod m

X2: (aX1 + c) mod m

.

M : 6

2,5,3,1,0,4,

Prof. Tanvi Goswami, DD University, Nadiad

a: 7, c=0, m=32, X0=1

Xn+1 = (a\* Xn + c) mod m

Full period?

Random?

Range is size of modulus

M 🡪 0 to 31

Only 4 values are produced

7, 17, 23, 1, 7, 17,23,1 so this generator is not full period, therefore we need to try new set of values for generating random numbers in PRNG

BLUM BLUM SHUB GENERATOR:

Cryptographically Secure Pseudo-Random Bit Generator (CSPRBG) 1) Choose 2 large prime number P & Q.

2) let n = P\*Q

3)Choose random value 🡪 seed (s) such that it is prime to “n” 4) ALGO:

X0 = S2 MOD n

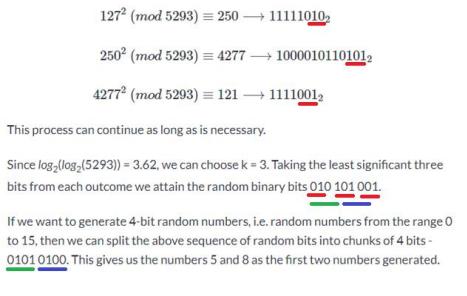
for i = 1 to inf

Xi = (Xi-1 )2 mod n

Prof. Tanvi Goswami, DD University, Nadiad

Bi = (Xi ) mod 2

As an example let p = 67 and q = 79 so that n = 67 × 79 = 5293. Let x0 = 127. Then the initial bits are calculated as follows.

 Prof. Tanvi Goswami, DD University, Nadiad