Tanuj Saini

2022511141

PAGE No.
DATE

1(d) We need to prove that second row of S_4 can be obtained from the first row of S_4 using the following mapping :-

$$(y_1, y_2, y_3, y_4) \rightarrow (y_2, y_1, y_4, y_3) \oplus (0, 1, 1, 0)$$

Lets check if 1st element of row 1 ($S_4[0][0]$) can be converted into 1st element of row 2 ($S_4[0][1]$)

$S_4[0][0] \rightarrow \underbrace{0\ 1\ 1\ 1}_{[7]_{10}} \rightarrow 1\ 0\ 1\ 1 \oplus 0\ 1\ 1\ 0$

$y_1\ y_2\ y_3\ y_4 \qquad y_2\ y_1\ y_4\ y_3$

$\rightarrow \underbrace{1\ 1\ 0\ 1}_{[13]_{10}} = S_4[1][0]$

→ Similarly lets check if $S_4[0][1]$ can be converted into $S_4[1][1]$ using the mapping

$S_4[0][1] = [3]_{10} \Rightarrow 1101 \Rightarrow 1110 \oplus 0110$

$\Rightarrow 1000 = [8]_{10} = S_4[1][1]$

• Now, check nesct element

$S_4[0][2] = [14]_{10} \Rightarrow 1110$

$\Rightarrow 1101 \oplus 0110$

$\Rightarrow 1011 = [11]_{10} = S_4[1][2]$

→ We will check this transformation for all the elements of row 1:

Tanuj Saini

2022571(4)

PAGE No.
DATE

$S_4[0][3] = [3]_{10} \Rightarrow 0011$

$\Rightarrow 0011 \oplus 0110$

$\Rightarrow 0101 = [5]_{10} = S_4[1][3]$

$\Rightarrow S_4[0][4] = [0]_{10} \Rightarrow 0000$

$\Rightarrow 0000 \oplus 0110 \Rightarrow 0110 = [6]_{10} = S_4[1][4]$

$\Rightarrow S_4[0][5] = [6]_{10} \Rightarrow 0110 \Rightarrow 1001 \oplus 0110$

$\Rightarrow 1111 = [15]_{10} = S_4[1][5]$

$\Rightarrow S_4[0][6] = [9]_{10} \Rightarrow 1001 \Rightarrow 0110 \oplus 0110$

$\Rightarrow 0000 = [0]_{10} = S_4[1][6]$

$\Rightarrow S_4[0][7] = [10]_{10} \Rightarrow 1010 \Rightarrow 0101 \oplus 0110$

$\Rightarrow 0011 = [3]_{10} = S_4[1][7]$

$\Rightarrow S_4[0][8] = [1]_{10} \Rightarrow 0001 \Rightarrow 0010 \oplus 0110$

$\Rightarrow 0100 = [4]_{10} = S_4[1][8]$

$\Rightarrow S_4[0][9] = [2]_{10} \Rightarrow 0010 \Rightarrow 0001 \oplus 0110$

$\Rightarrow 0111 = [7]_{10} = S_4[1][9]$

$\Rightarrow S_4[0][10] = [8]_{10} \Rightarrow 1000 \Rightarrow 1100 \oplus 0110$

$\Rightarrow 0010 = [2]_{10} = S_4[1][10]$

$\Rightarrow S_4[0][11] = [5]_{10} \Rightarrow 0101 \Rightarrow 1010 \oplus 0110$

$\Rightarrow 1100 = [12]_{10} = S_4[1][11]$

Tanuj Saini

2022571(M)

PAGE No.
DATE

→ $S_4[0][13] = [13]_{10} \Rightarrow 1100 \Rightarrow 1100 \oplus 0110$
$\Rightarrow 1010 = [10]_{10} = S_4[1][13]$

→ $S_4[0][14] = [4]_{10} \Rightarrow 0100 \Rightarrow 1000 \oplus 0110$
$\Rightarrow 1110 = [14]_{10} = S_4[13][14]$

→ $S_4[0][15] = [15]_{10} \Rightarrow 1111 \Rightarrow 1111 \oplus 0110$
$\Rightarrow 1001 = [9]_{10} = S_4[1][15]$

We proved that the entire second row of S_4 can be obtained from the first raw using the mentioned mapping. Hence, proved!

(b) Any row of S_4 can be transformed into any other row by a similar type of operation. Let us try to find mapping which will transform row 2 to row 4

Raw 2 $\xrightarrow{\text{mapping}}$ Row 4

$\underbrace{(y_1, y_2, y_3, y_4)}_{\text{Row 2}} \rightarrow \underbrace{(x_1, x_2, x_3, x_4)}_{\text{Row 4}}$

We will find correspondence from row 2 to 4 i.e, from (y_1, y_2, y_3, y_4) to (x_1, x_2, x_3, x_4)

Tanuj Saini

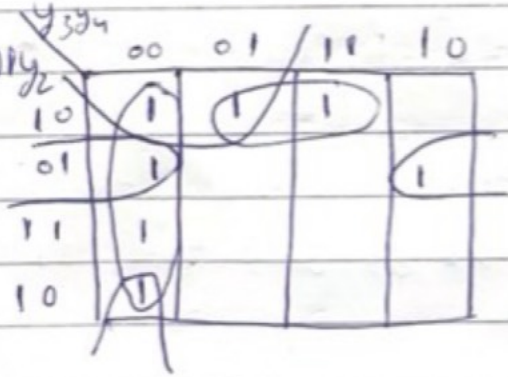
2022511411

PAGE No.
DATE

element	y_1	y_2	y_3	y_4	x_1	x_2	x_3	x_4	element
S[1][6]	0	0	0	0	1	1	0	1	S[3][6]
S[1][12]	0	0	0	1	1	1	0	0	S[3][12]
S[1][10]	0	0	1	0	0	1	0	1	S[3][10]
S[1][7]	0	0	1	1	1	0	0	0	S[3][7]
S[1][8]	0	1	0	0	1	0	0	1	S[3][8]
S[1][3]	0	1	0	1	0	1	1	0	S[3][3]
S[1][4]	0	1	1	0	1	0	1	0	S[3][4]
S[1][9]	0	1	1	1	0	1	0	0	S[3][9]
S[1][1]	1	0	0	0	1	1	1	1	S[3][1]
S[1][15]	1	0	0	1	1	1	1	0	S[3][15]
S[1][13]	1	0	1	0	0	1	1	1	S[3][13]
S[1][2]	1	0	1	1	0	0	0	0	S[3][2]
S[1][11]	1	1	0	0	1	0	1	1	S[3][11]
S[1][0]	1	1	0	1	0	0	1	1	S[3][0]
S[1][14]	1	1	1	0	0	0	1	0	S[3][14]
S[1][5]	1	1	1	1	0	0	0	1	S[3][5]

Now, we have the mapping. lets construct K-map in order to express x_i in terms y_i for all i

For x_1 :-



∴ + : Bitwise OR

• : Bitwise AND

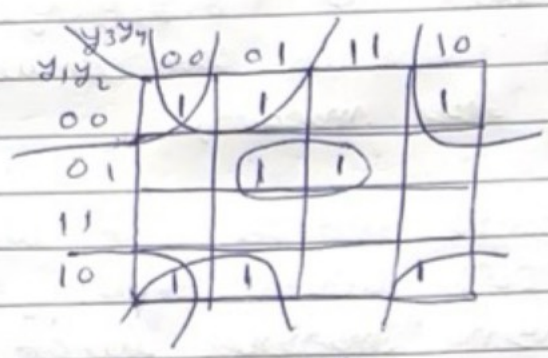
$(\bar{})$: complement

Tanuj Saini
2022511141

PAGE No.
DATE

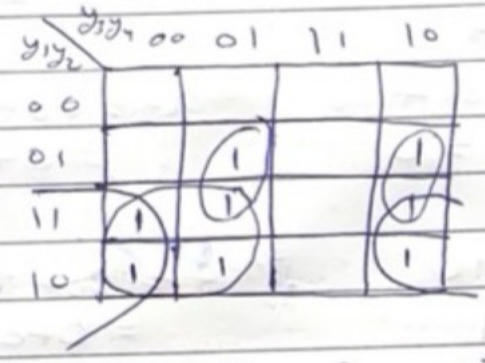
$x_1 = (\bar{y}_1 \bar{y}_2 y_4) + (y_1 \bar{y}_2 \bar{y}_4) + (\bar{y}_2 \bar{y}_3) + (\bar{y}_3 \bar{y}_4)$

→ For x_2 :-



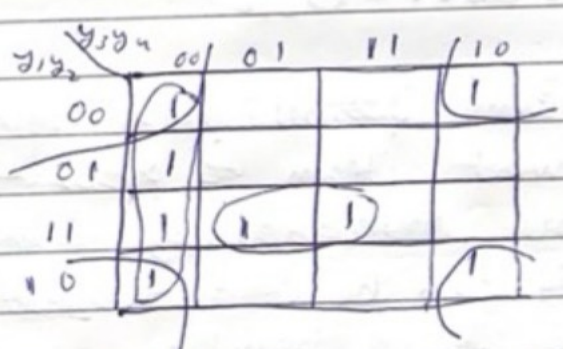
$x_2 = (\bar{y}_2 y_4) + (\bar{y}_1 y_2 y_4) + (y_2 \bar{y}_3)$

→ For x_3 :-



$x_3 = (y_2 \bar{y}_3 y_4) + (y_2 y_3 \bar{y}_4) + (y_1 \bar{y}_3) + (y_1 \bar{y}_4)$

→ For x_4 :-



$x_4 = (\bar{y}_2 \bar{y}_4) + (\bar{y}_3 \bar{y}_4) + (y_1 y_2 y_4)$

Tanuj Saini
2022511401

PAGE No.
DATE

We, therefore, successfully obtained mappings from Row 2 to Row 4 i.e, $(y_1 y_2 y_3 y_4)$ to $(x_1 x_2 x_3 x_4)$. Thus, we can convert row 2 to row 4 using these mappings.

Let us take an example and see if it works. Lets convert $S[1][0]$ which is $[13]_{10}$ to $S[3][0]$ which is $[3]_{10}$ using mappings we got

$$[13]_{10} = \underset{y_1\ y_2\ y_3\ y_4}{1\ 1\ 0\ 1}$$

$x_1 = 001 + 010 + 01 + 10$
$= 0 + 0 + 0 + 0 = 0$

$x_2 = 00 + 011 + 01$
$= 0 + 0 + 0 = 0$

$x_3 = 111 + 100 + 11 + 10$
$= 1 + 0 + 1 + 0 = 1$

$x_4 = 00 + 10 + 111$
$= 0 + 0 + 1 = 1$

$\therefore x_1 x_2 x_3 x_4 = 0011 = [3]_{10}$

We can verify the same for all other elements of row 2 and convert them to row 4. Now using this same methodology similarly, any row of S_4 can be transformed into any other row by a similar type of transformation.

Hence proved!

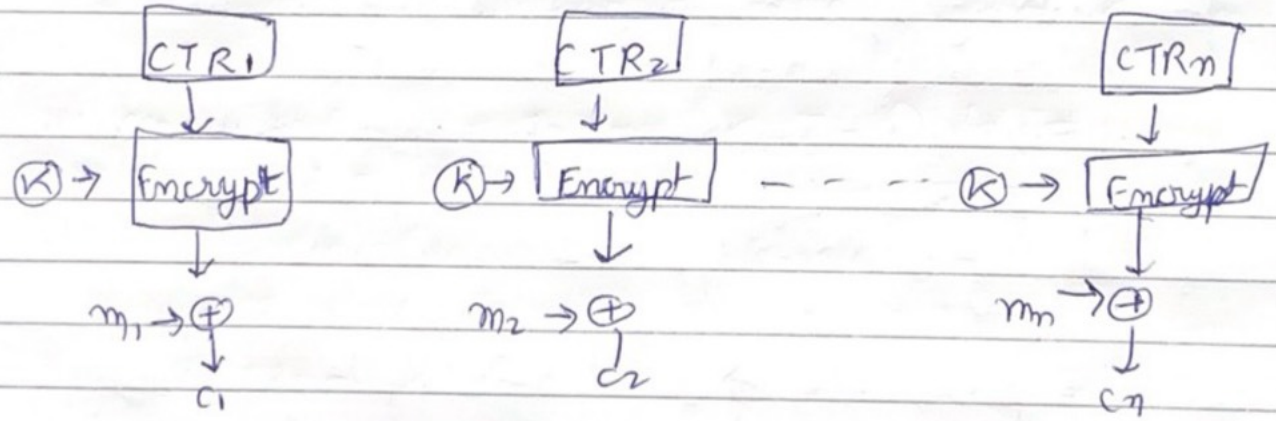
27 In CTR (counter) mode encryption, plain text is divided into blocks, and each block is encrypted separately using a block cipher (such as AES) with a unique counter value. Decryption in CTR mode is essentially the same process as encryption, just applied in reverse.

$P = m_1 || m_2 || m_2 - - - - || m_n$

Enc: $C_i = Enc_c \cdot (CTR_i, K) \oplus m_i$

$i = (0 \text{ to } n)$

Final ciphertext $C = C_1 || C_2 || - - - || C_n$

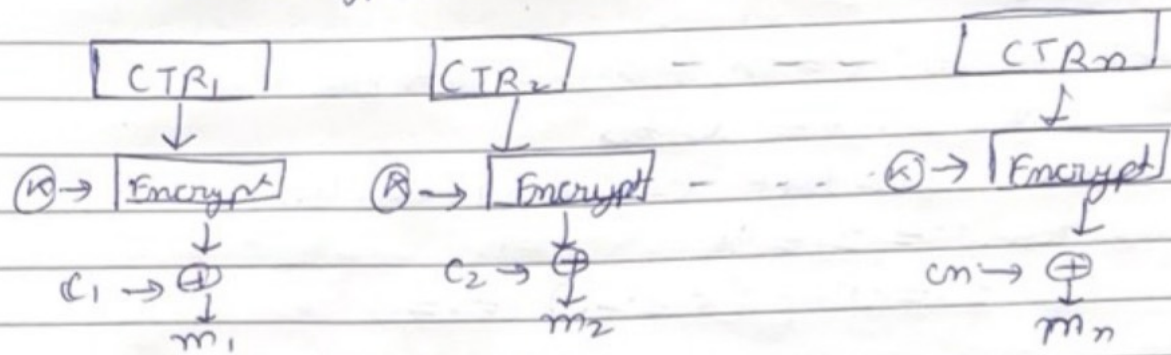


Tanuj Saini
2022571141

PAGE No.
DATE

Dec : $m_i = Enc(CTR_i \oplus K) \oplus C_i$
$i = (0 \text{ to } n)$

Final decryption $M = m_1 || m_2 - - - || m_n$



As, we can see from the block diagram of encryption and decryption, each block to be encrypted is independent. There is no dependency between the blocks which are getting encrypted. Each block can be encrypted independently and after all the blocks are encrypted, they can be arranged to get the final ciphertext. There is no chaining as in CBC. Hence, the process is efficiently parallelized.

The same is the case for decryption, there is no dependency between the blocks and hence each of the blocks can be decrypted independently. After that they can be arranged appropriately to get the final decrypted message. Hence, the process of decryption as well is efficiently parallelized.

Tanuj Saini
2022511141

PAGE No.
DATE

Q③ We have already understood CTR mode in the above question. let us now understand OFB mode (Output Feedback) of operation before proving what is asked in the question.

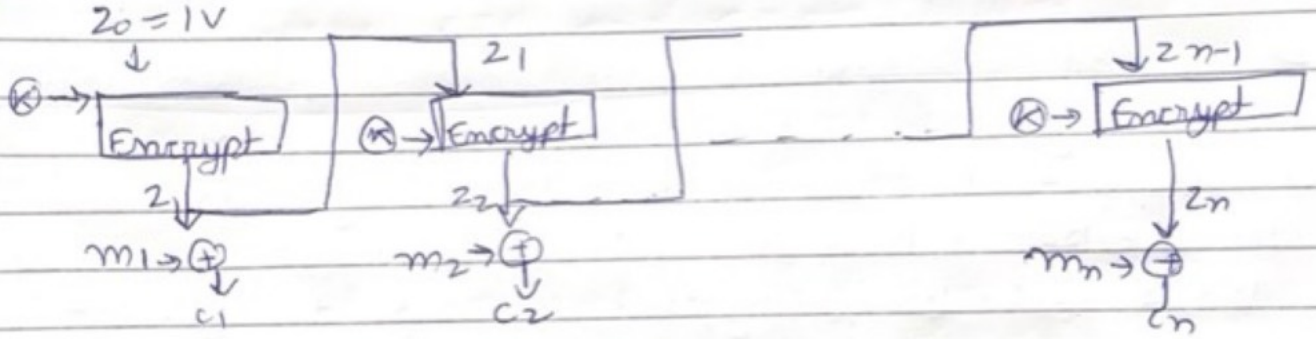
In OFB mode, $P = m_1 \| m_2 \| \cdots \| m_n$

IV is an inital vector

and final cipher text $C = C_1 \| C_2 \| \cdots \| C_n$

where $C_i = Z_i \oplus m_i$

and $Z_i = Enc(K, Z_{i-1}) \quad \forall i \neq 0$



Now, as given in the question, we have two sequences of n palintext blocks $X = (x_1, x_2, \cdots x_n)$ and $X' = (x_1', x_2', \cdots x_n')$. It is also given that both are encrypted using same K and IV

We know that $Z_i = Enc(K, Z_{i-1}) \quad \forall i \neq 0$ and if K and IV are same then $Z_i = Z_i'$ where $Z_i' = Enc(K, Z_{i-1}')$ because :-

$Z_0 = IV$	$Z_0' = IV$ (∵ same)
$Z_1 = Enc(K, IV)$	$Z_1' = Enc(K, IV)$
$Z_2 = Enc(K, Z_1)$	$Z_2' = Enc(K, Z_1')$

Tanuj Saini

2022571141

PAGE No.
DATE

Note that $z_0 = z_0'$, $z_1 = z_1'$, $z_2 = z_2'$ and
Hence $z_i = z_i'$ for $\forall i \neq 0$

Now, let consider C is cipher text corresponding to X where $C = (C_1, C_2, C_3, \ldots C_n)$ and C' is cipher text corresponding to X' where $C' = (C_1', C_2', \ldots C_n')$

$\therefore C = C_1 || C_2 || \ldots || C_n$

$C = (z_1 \oplus x_1) || (z_2 \oplus x_2) || \ldots || (z_n \oplus x_n)$

and $C' = (z_1 \oplus x_1') || (z_2 \oplus x_2') || \ldots || (z_n \oplus x_n')$

$C \oplus C' = [(z_1 \oplus x_1) \oplus (z_1 \oplus x_1')] || \ldots || [(z_n \oplus x_n) \oplus (z_n \oplus x_n')]$

$C \oplus C' = [x_1 \oplus x_1'] || [x_2 \oplus x_2'] || \ldots || [x_n \oplus x_n']$

$C \oplus C' = x_1 \oplus x_1' || x_2 \oplus x_2' || \ldots || x_n \oplus x_n'$

$= x \oplus x'$

Therefore, it is evident that $x \oplus x'$ can be computed easily given C and C' by doing $C \oplus C'$ and given X and X' are encrypted using OFB mode of operation using same K and IV.

Hence, proved

Similarly, if CTR mode is used

$C = C_1 || C_2 || \ldots || C_n$

$= Enc(CTR_1, K) = Enc(CTR_2, K) \ldots$

Lets call it z_i

Tanuj Saini

2022571141

PAGE No.
DATE

$\because C = z_1 \oplus x_1 \,||\, z_2 \oplus x_2 \,||\, \ldots \,||\, z_n \oplus x_n$

and $C' = z_1 \oplus x_1' \,||\, z_2 \oplus x_2' \,||\, \ldots \,||\, z_n \oplus x_n'$

$C \oplus C' = [(z_1 \oplus x_1) \oplus (z_1 \oplus x_1')] \,||\, \ldots \,||$

$C \oplus C' = x_1 \oplus x_1' \,||\, x_2 \oplus x_2' \,||\, \ldots \,||\, x_n \oplus x_n'$

$= X \oplus X'$

Therefore, it is evident that $X \oplus X'$ can be computed easily given C and C' key simply doing $C \oplus C'$ any given that CTR mode is used where CTR and K are reused.

Tanuj Saini
202251141

PAGE No.
DATE

4.) Linear Feedback shift Register (LFSR):
A sequential circuit that shifts bits with feedback determined by a connection polynomial. It's used for generating pseudorandom sequences, cryptography, and error correction.

Connection polynomial: Defines the feedback taps (positions of bits XORed together) for the LFSR.

Period: The number of steps required for the LFSR to return to its inital state.

Function to simulate the LFSR operation and then calculate the periods for the given connection polynomial

~~def lfsr(connection_polynomial):~~
int lfsr(unsigned int feedback_poly, int n){
unsigned int state = 1
unsigned int start_state = state;
int period = 0;

do{ unsigned int lsb = state & 1;
state >> = 1;
if (lsb){
state ^ = feedback_poly;
}
period ++; } while (state != start_state);
return period;
}

Tanuj Saini
2022511141

PAGE No. -
DATE

```
int main (){
    unsigned int feedback_a = 0b10011    // x^4+x+1
    int period_a = lfsr (feedback_a, 4);
    Printf ("LFSR f(x) = x^4+x+1 : %d \n", period_a);

    Unsigned int feedback_b = 0b100001;    // x^5+1
    int period_b = lfsr_period (feedback_b, 5);
    printf ("LFSR f(x) = x^5+1 : %d \n", period_b);
    return 0;
}
```

When we run the code

① period = 15
② period = 31

(a) $f(x) = x^4+x+1$

This polynomial indicates the feedback taps for the LFSR, In this case x^4, x^1, and the constant term 1. are the tap

→ The LFSR starts with an initial state of 0001
→ At each clock cycle, the rightmost bit is generated by XORing the taps: state[4] ⊕ state[1] ⊕ state[0]
→ Then all bits are shifted to right.

This process continues until the LFSR returns to its initial state, which occurs after 15 clock cycles, so, the period of this LFSR is 15.

Tanuj Saini

20225141

PAGE No.

DATE

(b) $f(x) = x^5 + 1$

constant term 1

, the LFSR starts with initial state of 00001

→ At each clock cycle, the rightmost bit is generated by XORing the taps:

state[5] ⊕ state[0]

· Then all bits are shifted to the right

So, the period of this LFSR is 31.

Tanuj Saini
2022571141

PAGE No.
DATE

5] $\lambda : Z_{105} \to Z_3 \times Z_5 \times Z_7$

$\lambda(x) = (x \bmod 3, x \bmod 5, x \bmod 7)$

We need to find λ^{-1} and hence compute $\lambda^{-1}(2,3,2)$

let $\lambda(x) = (a_1, a_2, a_3)$

and we are given $\lambda(x) = (x \bmod 3, x \bmod 5, x \bmod 7)$

$x \equiv a_1 \bmod 3 \quad x \equiv a_2 \bmod 5 \quad x \equiv a_3 \bmod 7$

Using chinese Remainder Theorem we can construct x such that

[A] $\to x = (a_1 m_1 m_1^{-1} + a_2 m_2 m_2^{-1} + a_3 m_3 m_3^{-1}) \bmod M$

and this x will satisfy all the 3 equations above

We are asked to find λ^{-1} which will take a_1, a_2, a_3 and give me x. Basically it will be a function from $Z_3 \times Z_5 \times Z_7 \Rightarrow Z_{105}$

In order to find λ^{-1} we need to solve equation [A]

$$M = 3 * 5 * 7 = 105$$

$$m_1 = \frac{105}{3} = 35 \qquad m_2 = \frac{105}{5} = 21$$

$$m_3 = \frac{105}{7} = 15$$

Now, m_1^{-1}, m_2^{-1} and m_3^{-1} will be calculated using extended euclidean algorithm.

Tanuj Saini
2022511411

PAGE No.
DATE

$\frac{M}{m_1} \cdot b_1 = 1 \mod (m_1)$

$35 \cdot \underbrace{b_1}_{m_1^{-1}} = 1 \mod (3)$

$\therefore m_1^{-1}$ is mul inverse of 35 under mod 3.

$3\overline{)35}\,(11$
-33
$2\overline{)3}\,(1$
-2
1

using extended euclided algorithm

$1 = 3 - 2 \times 1$

$1 = 3 - (35 - 3 \times 11)$

$1 = 12 \times 3 + 35 \times (-1)$

-1 under mod $3 = 2$

Now, for m_2^{-1} is mul inverse of 21 under mod 5

$5\overline{)21}\,(4$
20
1

$1 = 21 - 5 \times 4$

$1 = \underbrace{1 \times 21}_{21^{-1}} - 5 \times 4$

Now, for m_3^{-1} is mul inverse of 15 under mod 7

$7\overline{)15}\,(2$
-14
1

$1 = 15 - 7 \times 2$

$1 = \underbrace{1}_{15^{-1}} \times 15 - 7 \times 2$

$m_1^{-1} = 2$, $m_2^{-1} = 1$, $m_3^{-1} = 1$

$\therefore$ from eqation [*] we get

$x = (a_1 70 + a_2 21 + a_3 15) \mod 105$

Tanuj Saini

20225141

PAGE No.
DATE

3. $A^{-1} : Z_3 A 25 \times 2_7 \rightarrow Z105$

$A^{-1}(a_1, a_2, a_3) = (70a_1 + 21a_2 + 15a_3) \bmod 105$

Now, $A^{-1}(2,2,3) = [70(2) + 21(2) + 15(3)] \bmod 105$

$= (140 + 42 + 45) \bmod 105$

$= (227) \bmod 105$

$= 17$

$A^{-1}(2,2,3) = 17$

6) Product of all the moduli

$M = 25 \cdot 26 \cdot 27$

$M = 17550$

For each congruency, Find the value of $M_i = \frac{M}{m_i}$

$M_1 = \frac{17550}{25} = 702$

Tanuj Saini
20225114

PAGE No.
DATE

$M_2 = \frac{17550}{26} = 674.99$

$M_3 = \frac{17550}{27} = 650.74$

For each congruence find the modular inverse of m_i modulo m_i denoted by a_i

$a_i = m_i^{-1} \pmod{m_i}$ {Extended Euclidean Algorithm}

$702 = 0 \pmod{25}$

$702 = 1 \pmod{7}$ $\quad 702 = 7 \cdot 100 + 2$

$7 = -2 \pmod{9}$ $\quad 7 = 9 - 2$

$702 = 1 (-2) (100) \bmod 9$

$a_1 = -200 \pmod{9} = 7 \bmod 8$

$a_2 = -15 \pmod{26} = 11 \bmod (26)$

$a_3 = 23 \bmod (27)$

$i = 3$

$x = \sum_{i=1} (a_i \cdot m_i \cdot x_i) \text{ modulo } M$

$x = (a_1 \cdot m_1 \cdot 12 + a_2 \cdot m_2 \cdot 9 + a_3 \cdot m_3 \cdot 23) \pmod{M}$

$x = (7 \cdot 702 \cdot 12 + 11 \cdot 674.99 \cdot 9 + 8 \cdot 650.74 \cdot 23) \bmod (17550)$

$x = 2353476 \bmod (17550)$

$x = 1427 \bmod (17550)$

$x = 1427 \bmod (17550)$

Tanuj Saini

2022511141

PAGE No.
DATE

7) $n = 18923$ encryption key $e = 1261$

ciphertext $c = 6127$

$n = 127 \cdot 149$

Clearly n, decomposes into the product of two prime numbers

RSA encryption uses large prime numbers and modular arithmetic. Here we are able to decompose n into the product of two prime numbers easily because the numbers are small. If the number were very large then this would be a very complicated process.

Euler's ϕ function for n. Since n is the product of two prime

$\phi(n) = \phi(18923) = \phi(127 \cdot 149)$

$= (127-1)(149-1) = 18648$

$\phi(n) = (p-1)(q-1)$

Now we compute the private key d which is given by

$d = e^{-1} \pmod{\phi(n)}$

$= 1261^{-1} \pmod{18648}$

$= 5797 \pmod{18648}$

$5797 \cdot 1261 \equiv 1 \pmod{18648}$

c = ciphertext

$P = c^d \pmod{n}$

$= 6127^{5797} \pmod{18923}$

$= 5746 \pmod{18923}$

Tanuj Saini

2022511411

PAGE No.
DATE

Hence, for the RSA cryptosystem with $n = 18923$, $e = 12721$ when the ciphertext is $C = 6127$, the corresponding ans ~~plaintext is p~~ as follow. $x = (6127)^{5797} \bmod 18923$

→ To efficiently calculate $(6127)^{5797}$ we will use square-and-multiply algorithm ∵ 5797 in binary is: $[1011010100101]_2$

→ In square and multiply algo, whenever we encounter 1, in binary rep of 5797, we will multiply and whenever we encounter 0, we will square it simply. here $C = 6127$ $n = 18923$

1) $c^1 \cdot c^1 = (6127)^2 \bmod n = 15820$
2) $c^{10} \cdot c^{10} = (15820)^2 \bmod n = 15725$
3) $c^{101} = (15725 \times 6127) \bmod n = 10082$
4) $c^{1010} = (10082)^2 \bmod n = 11291$
5) $c^{1011} = (11291 \times 6127) \bmod n = 16392$
6) $c^{10110} = (16392)^2 \bmod n = 9987$
7) $c^{101100} = (9987)^2 \bmod n = 15959$
8) $c^{101101} = (15959 \times 6127) \bmod n = 5652$
9) $c^{1011010} = (5652)^2 \bmod n = 3080$
10) $c^{10110100} = (3080)^2 \bmod n = 5977$
11) $c^{10110101} = (5977 \times 6127) \bmod n = 5074$
12) $c^{101101010} = (5074)^2 \bmod n = 10196$
13) $c^{1011010100} = (10196)^2 \bmod n = 14377$
14) $c^{10110101000} = (14377)^2 \bmod n = 2200$
15) $c^{10110101001} = (2200 \times 6127) \bmod n = 6224$
16) $c^{10110101 0010} = (6224)^2 \bmod n = 2795$
17) $c^{101101010 0100} = (2795)^2 \bmod n = 15749$
18) $c^{1011 0101 0010 1} = (15749 \times 6127) \bmod n = 5746$

$\therefore x = c^d \bmod n = 5746$

$\therefore$ Plaintext is 5746

Tanuj Saini
2022511141

PAGE No.
DATE

5) $EL: y^2 = x^3 + 5x + 3$ over Z_{13}.

We wish to find all possible points on EL.

Here x in b/w $[0, 12]$ and y is between $[0, 12]$

A point (x, y) lies on curve if it satisfies the equation over mod 13. Basically LHS = RHS

where LHS $= y^2 \bmod 13$ RHS $= x^2 + 5x + 3 \bmod 13$

Let us compare a table to store these LHS/RHS value

LHS

y	$y^2 \bmod 13$
0	0
1	1
2	4
3	9
4	3
5	12
6	10
7	10
8	12
9	3
10	9
11	4
12	1

RHS

x	$x^2 + 5x + 3 \bmod 13$
0	3
1	9
2	8
3	6
4	9
5	10
6	2
7	4
8	9
9	10
10	0
11	11
12	10

Now, we will consider all possible values of (x, y) where LHS = RHS

$(0,4), (0,9), (1,3), (1,10), (4,3), (4,10), (5,6), (5,7), (7,2), (7,11), (8,3), (8,10), (9,6), (9,7), (10,0), (12,6), (12,7)$

Therefore, there are 17 such points on the given EL.

Tanuj Saini

2022511411

PAGE No.
DATE

9) In the scenario where the S-box denoted as π_t is not a permutation and specifically not surjective, it implies that π_t does not map to all possible values in its range. This characteristic can be exploited in a ciphertext-only attack to deduce key bits used in the last round of the SPN (Substitution-permutation Network).

Non-Surjectivity of π_t: Since π_t is not surjective, there exist some values in the output space that are never produced by π_t. Identify these values by analyzing the output range π_t.

→ Collection of ciphertexts: Collect a large number of ciphertexts encrypted under the same key. The more ciphertexts gathered, the higher the probability of covering a broad range of π_t output.

→ Analysis of Last Round Outputs: For each ciphertext, partially decrypt it by reversing the last round's permutation (if any) and then apply the inverse of the last round key (guessing the key bits). This step is possible because the permutation layer is typically a known, fixed operation, and the key space for the last round might be small enough to allow exhaustive search.

Tanuj Saini
2022511411

→ Identification of Impossible Outputs :- check if the output from the decryption step above results in a value that it could not have produced. If such a value is found, the guessed key bits for the last round are incorrect.

→ Key Elimination: Repeat step 3 with different guesses for the key bits. Each incorrect key guess will likely result is an impossible output for n ∈. The correct key bits do not lead to such an output.

10) Given, a hash function $h : Z_n \times Z_n \to Z_n$ which is a linear function $h(x,y) = ax + by \bmod n$ where $a, b \in Z_n$ and $n \geq 2$.

Let say we are given the value :-

$h(x_1, y_1) = z_1$

$h(x_2, y_2) = z_2$

Let $r, s \in Z_n$ then we have that

$= h(rx_1 + sx_2 \bmod n,\ ry_1 + sy_2 \bmod n)$

$= a(rx_1 + sx_2) + b(ry_1 + sy_2) \bmod n$

$= r(ax_1 + by_1) + s(ax_2 + by_2) \bmod n$

$= r\,h(x_1, y_1) + s\,h(x_2, y_2) \bmod n$

$= rz_1 + sz_2$

Tanuj Saini

2022511141

Therfore, given the value of function h at two points (x_1, y_1) and (x_2, y_2), we know its value at various other points, in this case at $(ax_1 + bx_2 \bmod n, ay_1 + by_2 \bmod n$. This was done without actually having to evaluate h at those points (and also note that we do not even to know the values of constants a and b in order to apply the above-described technique. Hence, proved!

ii) $p \rightarrow$ prime number

$a, b \in Z_p$

$f(a,b) : Z_p \rightarrow Z_p$ by $f(a,b)(x) = ax + b \bmod p$

$x \neq x' \in Z_p$ such that & $f(a,b)(x) = y$

$f(a,b)(x') = y'$

$(ax + b) \equiv y \bmod p$ — (A)

$(ax' + b) \equiv y' \bmod p$ — (B)

(A) - (B)

$a(x - x') = (y - y') \bmod p$

we know, $x - x' \neq 0 \because x \neq x' \rightarrow$ given

So, for checking if we can find a and b, we need to check if $(x - x')^{-1}$ exists or not. i.e if $(x - x^{-1})$ is invertible or not under mod p. inverse of $(x - x')$ mod p exists if and gcd $(x - x', p) \equiv 1$

As p → prime No, → gcd $(x - x', p) = 1$ always

Tanuj Saini

2022511141

PAGE No.

DATE

∴ Inverse of $(x - x')$ mod p exists.

So, we can find 'a' and 'b'.

for a :- $a \equiv (y - y')(x - x')^{-1} \bmod p$ — (C)

for b :- After solving equation (1), put value of 'i' obtained in equation (A) and (B) to find 'b'.

Hence, we can derive a and b