Name : Pradeep Rekapalli Date : 19-09-2024

KSU ID : 001100818

NETID : PREKAPAL

CS 7530 Fall 2024

Assignment-2

Question-1

① Calculate $\phi(230)$.

Given to Calculate $\phi(n)$ [Euler's function] which counts the number of integer less than or equal to n. that are relatively prime (co) to n.

To so inorder to Calculat $\phi(230)$ by this steps.

Step1: we need find prime factorization of 230.

which is $2 \mid 230$, $5 \mid 115$, 23 So $230 = 2 \times 5 \times 23$ (P_1 P_2 P_3)

Step 2: Euler's function formula

$$\phi(n) = n \times \left(1-\frac{1}{P_1}\right) \times \left(1-\frac{1}{P_2}\right) \times \cdots \times \left(1-\frac{1}{P_k}\right)$$

So now we substitude this & find

$$\phi(230) = 230 \times \left(1-\frac{1}{2}\right) \times \left(1-\frac{1}{5}\right) \times \left(1-\frac{1}{23}\right)$$

$$\phi(230) = \cancel{230}^{115} \times \frac{1}{\cancel{2}} \times \frac{\cancel{4}}{\cancel{5}} \times \frac{22}{\cancel{23}}$$

So $\phi(230) = 88$

00 11 00 818

(2)

(2) using euler's theorem calculate

Ans

$a^{\phi(n)+1} = 27^{89} \bmod 230$ $\quad n = 230$, $a = 27$

before doing this but before we already calculate

$\phi(230) = 88$ so let

$27^{\phi(230)+1} = 27^{89} \bmod 230$ → Equation

LHS RHS

~~$27^{88+1} = 27^{89} \bmod 230$~~

according to theorem it is $a^{\phi(n)+1} = 1 \bmod n$.

Here 27, 230 are coprime.

according to theorem $a^{\phi(n)} = 1 \bmod 230$

(1) ← $27^{88} = 1 \bmod 230$

Substitute it in equation:

001100818

(3)

$27^{89} \bmod 230$

$27 \times 27^{88} \bmod 230 \longrightarrow$ from (1)

$27 \times (1)$ so

$27^{89} \bmod 230 = [27]$

Section 2 (Question)

(a) Ans

Given to encrypt the plaintext "cryptography" using the Vigenere Cipher Basics:

So we need to shift by the corresponding number in the key stream.

So for example a key shift of 9 shifts the first letter of the plaintext by 9 positions in the alphabet.

So

Given:

* plaintext = "cryptography".

Key stream = 9, 11, 15, 2, 8, 16, 1, 21, 16, 18, 13, 7

So first we need to convert alphabets we need to know its positions when

So A=0, B=1, ~~~~ Z=25 So

c=2, r=17, y=24, p=15, t=19, o=14, g=6, r=17, a=0,

p=15, h=7, y=24. [** if we get >26 we do mod]

So Now

c → 2+9 = 11 → L

r → 17+11 = 28 mod 26 = 2 → C

y → 24+15 = 39 mod 26 = 13 → N

p → 15+2 = 17 → R

t → 19+8 = 27 mod 26 = 1 → B

a → 14+16 = 30 mod 26 = 4 → E

g = 6+1 = 7 → H

r = 17+21 = 38 mod 26 = 12 → M

a = 0+16 = 16 → Q

~~Pf = 15+18~~

P = 15+18 = 33 mod 26 = 7 → H

h = 7+13 = 20 → U

y = 24+7 = 31 mod 26 = 5 → F

So now ciphertext = "

= "LCNRBEHMQHUF"

001100818
(5)

(13) Ans So ciphertext = "LCNRBEHMQHUF" → (1)

Desired Plaintext = "kkarkennesawuniw" → (2)

So Now we need to do (1) − (2) to find key

If we get a

[If we get a number <0 or >26 we do mod]

1. L(11) − k(10) = 1 → key = (1) → key
2. C(2) − e(4) = −2 ≡ −2 mod 26 = 24
3. N(13) − n(13) = 0 = 0
4. R(17) − n(13) = 4 = 4
5. B(1) − e(4) = −3 mod 26 = 23
6. E(4) − S(18) = −14 mod 26 = 12
7. H(7) − a(0) = 7
8. M(12) − w(22) = −10 = +10 mod 26 = 16
9. Q(16) − u(20) = −4 mod 26 = 22
10. H(7) − n(13) = −6 mod 26 = 20
11. U(20) − i(8) = 12
12. F(5) − v(21) = −16 mod 26 = 10

So the key is

Key = 1, 24, 0, 4, 23, 12, 7, 16, 22, 20, 12, 10

Part 2 : Question 3:

Report:

As in question it is given to implement DES Function, and we need to give two inputs Ri(32 bits) and Ki(48bits) which produces a 32 bit output.

First lets understand DES (Data Encryption Standard). Here the F Function is main part.as f-fucntion need to give two inputs Ri(32 bit) and Ki (48 bit) here we will do Ri to 48 bits by using expansion which I will explain in steps. And then XOR it with Ki then finally permuate it to get 32 bit ouput.

So here I am dividing whole procedure into few sections for detailed explanation.I have written sections based on the procedure.

1. Step:Expansion;
   Here we need to expand Ri 32 bit to 48 bits by using expansion table
   It generally expandeds bots by repreatimg the certain bits from RI
   Here we have done in expand function
2. Step:XOR:
   The 48 bit is then XOR by the 48 bit round key Ki.
3. Step:Subsititution (S box):
   Here after xor oprtation the 48 bit ones are divided into 6 bit blocks
   Each 6 bits are then converted by using S-box like in 6 bits , 1^{st} and 6^{th} bit are determine the rows and inside 4 bits decide column
   So that we can say that each Sbox output is 6 bit input to 4 bit putput
4. Step:Permutation (P box)
   So total again sum 32 bits are formed as we are excluding 2 from 6 bits.and these are permeated through but using permutation table
   Here during the this procedure we reaarange ouputs for final output.This is key procedure.

Additional :

Here I used hex_to_bin and bin_to_hex to convert the hexadecimal input to binary and vide versa which I took reference from greeksforgreeks. And I have implemented code in C++.

For code I have took reference from and lectures provided

Reference : https://github.com/alimasry/DES-Encryption/

Code :

```
#include <iostream>
#include <string>
#include <bitset>
#include <sstream>
#include <iomanip>

using namespace std;

const int expansion_table[48] = {
   32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9,
   8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
   16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,
   24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1
};

const int permutation_table[32] = {
   16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26,
   5, 18, 31, 10, 2, 8, 24, 14, 32, 27, 3, 9,
   19, 13, 30, 6, 22, 11, 4, 25
};

const int sbox[8][4][16] = {
   {{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7},
    {0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8},
    {4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0},
    {15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13}},
   // S2-S8 omitted for brevity. Include all 8 S-boxes in a real implementation.
```

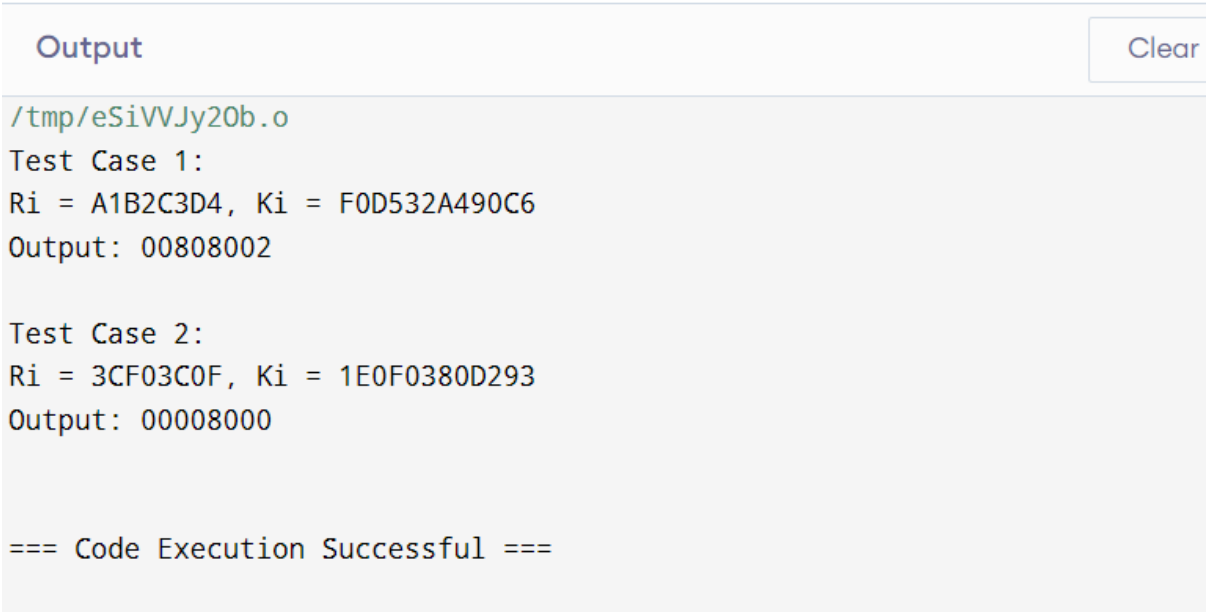
```
};

string hex_to_bin(const string& hex) {
    string bin;
    for (char c : hex) {
        int n = (c >= 'A') ? (c - 'A' + 10) : (c - '0');
        bin += bitset<4>(n).to_string();
    }
    return bin;
}

string bin_to_hex(const string& bin) {
    stringstream ss;
    for (size_t i = 0; i < bin.length(); i += 4) {
        string chunk = bin.substr(i, 4);
        int decimal = stoi(chunk, nullptr, 2);
        ss << hex << uppercase << decimal;
    }
    return ss.str();
}

string expand(const string& block) {
    string expanded;
    for (int i : expansion_table) {
        expanded += block[i - 1];
    }
    return expanded;
}

string xor_strings(const string& a, const string& b) {
    string result;
```

```
    for (size_t i = 0; i < a.length(); ++i) {
        result += (a[i] == b[i]) ? '0' : '1';
    }
    return result;
}

string apply_sbox(const string& block) {
    string output;
    for (int i = 0; i < 8; ++i) {
        int row = 2 * (block[i*6] - '0') + (block[i*6 + 5] - '0');
        int col = 8 * (block[i*6 + 1] - '0') + 4 * (block[i*6 + 2] - '0') +
                  2 * (block[i*6 + 3] - '0') + (block[i*6 + 4] - '0');
        int val = sbox[i][row][col];
        output += bitset<4>(val).to_string();
    }
    return output;
}

string permute(const string& block) {
    string permuted;
    for (int i : permutation_table) {
        permuted += block[i - 1];
    }
    return permuted;
}

string f_function(const string& Ri, const string& Ki) {
    string expanded = expand(Ri);
    string xored = xor_strings(expanded, Ki);
    string substituted = apply_sbox(xored);
    string permuted = permute(substituted);
```

```
    return permuted;
}

int main() {
    // Test case 1
    string Ri_hex1 = "A1B2C3D4";
    string Ki_hex1 = "F0D532A490C6";

    string Ri_bin1 = hex_to_bin(Ri_hex1);
    string Ki_bin1 = hex_to_bin(Ki_hex1);

    string result_bin1 = f_function(Ri_bin1, Ki_bin1);
    string result_hex1 = bin_to_hex(result_bin1);

    cout << "Test Case 1:" << endl;
    cout << "Ri = " << Ri_hex1 << ", Ki = " << Ki_hex1 << endl;
    cout << "Output: " << result_hex1 << endl << endl;

    // Test case 2
    string Ri_hex2 = "3CF03C0F";
    string Ki_hex2 = "1E0F0380D293";

    string Ri_bin2 = hex_to_bin(Ri_hex2);
    string Ki_bin2 = hex_to_bin(Ki_hex2);

    string result_bin2 = f_function(Ri_bin2, Ki_bin2);
    string result_hex2 = bin_to_hex(result_bin2);

    cout << "Test Case 2:" << endl;
    cout << "Ri = " << Ri_hex2 << ", Ki = " << Ki_hex2 << endl;
    cout << "Output: " << result_hex2 << endl;
```

```
    return 0;
}
```

Output:



Here is output, including test case. Professor, I am in hackthon at Morgan Stanley, so if anything needs to be done additionally, I missed anything. I would request for again submission after the deadline.

Thank you

Pradeep Rekapalli

KSU ID : 001100818

NET ID : prekapal