CSCI 6708 NETWORK SECURITY

ASSIGNMENT - 3

1. **Program to encrypt and decrypt strings**

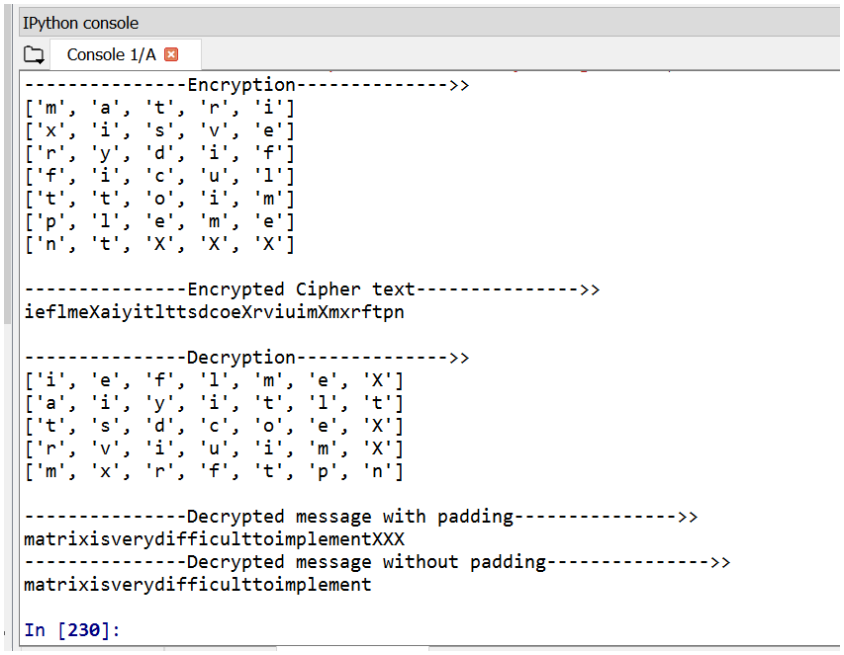
a) **Caeser Cipher (Python Code attached)**

```
In [205]: runfile('C:/Users/Yamuna/Desktop/test.py',
wdir='C:/Users/Yamuna/Desktop')

Enter message to be encrypted:Mark Zuckerberg

Enter key for Ceaser Cipher:5

Encryption using Ceaser Cipher-------->>
Rfwp%Ezhpjwgjwl

Decryption using Ceaser Cipher-------->>
Mark Zuckerberg
```

b) **Vigenere Cipher (Python Code attached)**

```
In [230]: runfile('C:/Users/Yamuna/Desktop/Network
Security/Assignment3/Vignere.py',
wdir='C:/Users/Yamuna/Desktop/Network Security/Assignment3')

Encryption using Vigenere Cipher-------->>
Key Mapping
[('M', 'f'), ('a', 'a'), ('r', 'c'), ('k', 'e'), ('Z', 'b'),
('u', 'o'), ('c', 'o'), ('k', 'k'), ('e', 'f'), ('r', 'a'),
('b', 'c'), ('e', 'e'), ('r', 'b'), ('g', 'o')]

Encrypted Result-------->>
RatoAiqujrdisu

Decryption using Vigenere Cipher-------->>
Key Mapping
[('R', 'f'), ('a', 'a'), ('t', 'c'), ('o', 'e'), ('A', 'b'),
('i', 'o'), ('q', 'o'), ('u', 'k'), ('j', 'f'), ('r', 'a'),
('d', 'c'), ('i', 'e'), ('s', 'b'), ('u', 'o')]

Decrypted Result-------->>
MarkZuckerberg
```

c) **Matrix Transposition Cipher (Python Code attached)**



2. AES Introduction

One of the most popular and widely accepted symmetric encryption algorithm expected to run into these days is the Advanced Encryption Standard (AES). It is proved to be at least six time faster than triple DES.

A replacement for the traditional DES algorithm was required as its key size was too small. With growing calculating power, it was measured defenceless against comprehensive key search attack. Triple-DES was intended to overcome this downside but it was found slow.

Some of the features of AES algorithm are as follows −

- It uses symmetric key and symmetric block cipher
- It uses 128-bit data with 128/192/256-bit keys
- It is stronger and much faster than Triple-DES
- It provides full specification and design details
- It can be implemented in software using C and Java

Operation of AES

AES is iterative compared to than Feistel cipher. It is based on logic substitution and permutation network. It includes of a series of inter-linked processes, some of which involve swapping inputs by specific outputs (substitutions) and some others involve shuffling bits around (permutations).

Fascinatingly, AES achieves all its computations in bytes rather than in bits. Henceforth, AES indulgences the 128 bits of a plain text block as 16 byte block. These 16 bytes are organized in four columns and four rows for dispensation as a matrix

Contrasting DES, the number of rounds in AES is inconstant and be contingent on the length of the key. AES practices 10 routines for 128-bit keys, 12 routines for 192-bit keys and 14

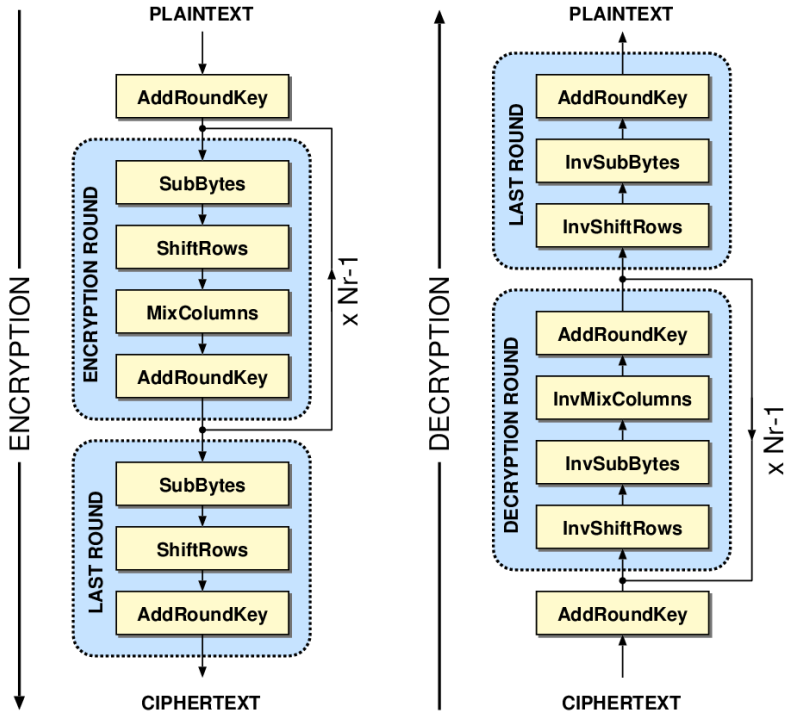
routines for 256-bit keys. Each of these routines uses a changed 128-bit round key, which is computed from the original AES key.

Key Generation Process

The cipher key used for encryption is 128 bits long. The cipher key is by now the outcome of many hashing and cryptographic alterations and, by the time it reaches at the AES block encryption, it is far detached from the secret master key detained by the authentication server. Now, finally, it is used to produce a set of eleven 128-bit round keys that will be united with the message during encryption. Though there are ten rounds, eleven keys are chosen because one extra key is added to the early state array before the rounds start.

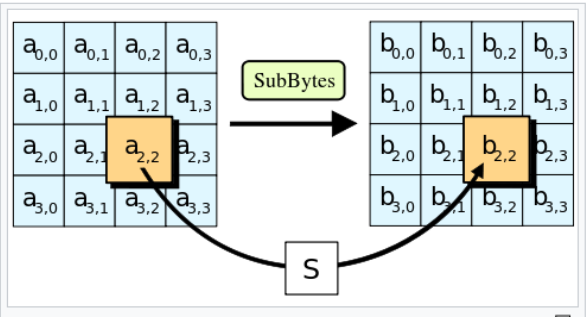
Encryption Process

The description is restricted to a single routine of AES encryption. Each routine encompasses four sub-processes. The first sub-process is portrayed below.



Byte Substitution (Sub Bytes)

The input to this sub-process is 16 bytes that are substituted by a look-up process on a table called S-box. The outcome is in a matrix of four rows and four columns. Image below depicts the process diagrammatically

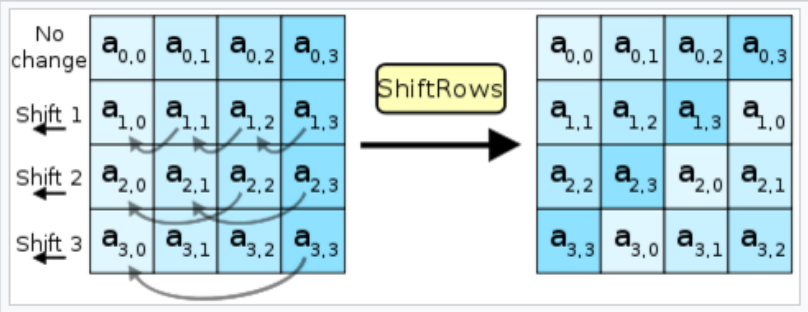


Shift rows

Among the four rows of the matrix, each of it is shifted to the left. The remaining entries that fall off will be re-inserted on the right side of the row. Shift is passed out as follows

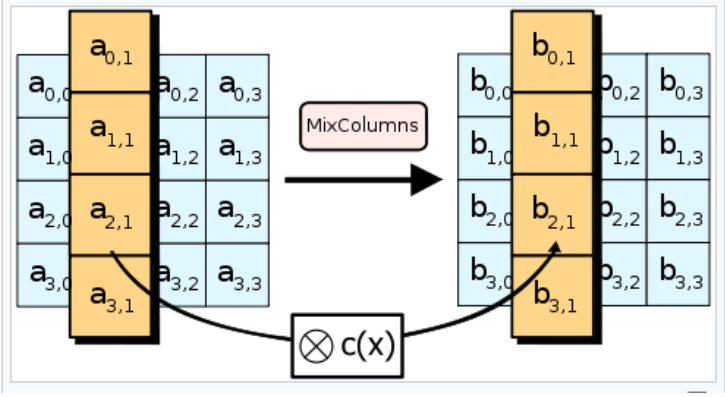
- First row should not be shifted.
- Second row will be shifted one position or byte to the left.
- Third row will be shifted two positions to the left.
- Fourth row will be shifted three positions to the left.
- The outcome is a new matrix containing the same 16 bytes but shuffled with respect to each other.

Image below depicts the process diagrammatically



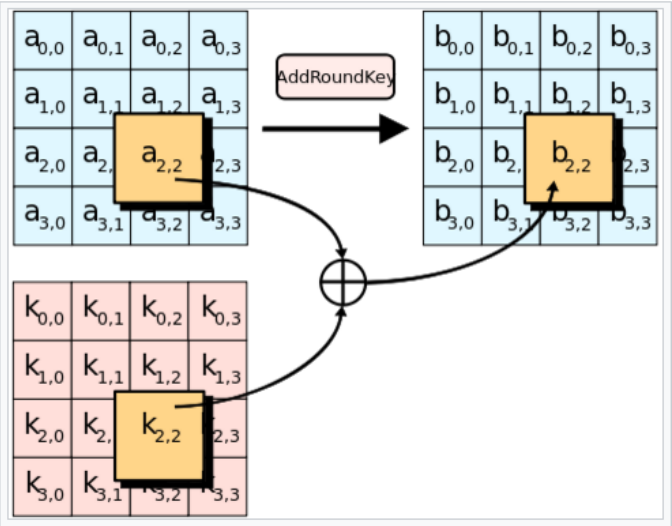
Mix Columns

Each of the column on the four bytes will be altered using a special scientific function. This function will take as input the four bytes of a single column and substitutes with four entirely new bytes, which exchange the original column. The outcome is another completely new matrix containing of 16 new bytes. It is to be noted that this step is not achieved in the last routine. Image below depicts the process diagrammatically



Add round key

The given 16 bytes of the matrix is now taken as 128 bits instead of bytes and XOR is performed with the 128 bits of the routine key. If this is considered to be the last round, then the outcoming text is the cipher text. Or, the resultant 128 bits are taken as 16 bytes and this process is iterated again. Image below depicts the process diagrammatically



Decryption Process

The decryption process of an encrypted AES cipher text is almost similar to the encryption process but in the reverse order. Each routine contains of the four processes mentioned below computed in reverse order

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each routine are in inverse manner, contrasting Feistel Cipher, the encryption algorithm and the decryption algorithm should be implemented separately, though they are very closely related.

AES Analysis

AES is widely implemented and supported in present day cryptography in both hardware and software. Until today, no practical cryptanalytic attacks against AES has been discovered. Furthermore, AES has in-built flexibility for key length, which permits a degree of 'future-proofing' against development in the ability to perform comprehensive key searches. However, just as for DES, the AES security is guaranteed only if it is correctly applied and good key management is employed.

3. RSA Problem

(a) p = 7, q = 11, M = 6

1. p = 7, q = 11

2. n = p*q

3. n = 7*11 = n = 77

4. e = 7

5. ed mod (p-1) (q-1) = 1

7d mod (7-1) (11-1) = 1

7d mod 60 = 1

With possible values for 7d - 61, 121, 181, 241, 301, using trial and error

7d = 301

d = 43

Public Key (e, n) = (7, 77)

Public Key (d, n) = (43, 77)

Encrypted Cipher Text:

c = m^e mod n

c = 6^7 mod 77

c = (6^3*6^3*6) mod 77

c = ((6^3 mod 77) * (6^3 mod 77) * (6 mod 77)) mod 77

c = ((62) * (62) * 6) mod 77

c = 3968 mod 77

Encrypted text: c = 41

(b) p = 11, q = 13, M = 9

1. p = 11, q = 13

2. n = p*q

3. n = 11*13 = n = 143

4. e = 7

5. ed mod (p-1) (q-1) = 1

7d mod (11 - 1) (13 - 1) = 1

7d mod 120 = 1

With possible values for 7d - 121, 241, 361, 481, 601, 721, using trial and error

7d = 721

d = 103

Public Key (e, n) = (7, 143)

Public Key (d, n) = (103, 143)

Encrypted Cipher Text:

c = m^e mod n

c = 9^7 mod 143

c = (9^3*9^3*9) mod 143

c = ((9^3 mod 143) * (9^3 mod 143) * (9 mod 143)) mod 143

c = ((14) * (14) * 9) mod 143

c = ((196 mod 143) * 9) mod 143

c = (53 * 9) mod 143

c = 477 mod 143

Encrypted text: c = 48

(c) p = 17, q = 31, M = 5

1. p = 17, q = 31

2. n = p*q

3. n = 17 * 31 = n = 527

4. e = 7

5. ed mod (p-1) (q-1) = 1

7d mod (17 - 1) (31 - 1) = 1

7d mod 480 = 1

With possible values for 7d - 481, 961, 1441, 1921, 2401

Using trial and error, 7d = 2401

d = 343

Public Key (e, n) = (7, 527)

Public Key (d, n) = (343, 527)

Encrypted Cipher Text:

c = m^e mod n

c = 5^7 mod 527

c = (5^4*5^3) mod 527

c = ((5^4 mod 527) * (5^3 mod 527)) mod 527

c = ((98) * (125)) mod 527

c = 12250 mod 527

Encrypted text: c = 129

4. RSA Algorithm

(a) p = 7, q = 11, M = 6

```
In [124]: runfile('C:/Users/Yamuna/Desktop/array.py',
wdir='C:/Users/Yamuna/Desktop')

Enter value of 'p': 7

Enter value of 'q': 11

Enter Message to be encrypted: 6
Public Key (e,n): ( 7 , 77 )
Private Key (d,n): ( 43 , 77 )
6
Encrypted cipher text : 41
Decrypted plain text : 6

Encryption and decryption using Lecture notes formula

Encrypted cipher text : 41
Decrypted plain text : 6
```

(b) p = 11, q = 13, M = 9

```
In [126]: runfile('C:/Users/Yamuna/Desktop/array.py',
wdir='C:/Users/Yamuna/Desktop')

Enter value of 'p': 11

Enter value of 'q': 13

Enter Message to be encrypted: 9
Public Key (e,n): ( 7 , 143 )
Private Key (d,n): ( 103 , 143 )
9
Encrypted cipher text : 48
Decrypted plain text : 9

Encryption and decryption using Lecture notes formula

Encrypted cipher text : 48
Decrypted plain text : 9
```

(c) p = 17, q = 31, M = 5

```
In [127]: runfile('C:/Users/Yamuna/Desktop/array.py',
wdir='C:/Users/Yamuna/Desktop')

Enter value of 'p': 17

Enter value of 'q': 31

Enter Message to be encrypted: 5
Public Key (e,n): ( 7 , 527 )
Private Key (d,n): ( 343 , 527 )
5
Encrypted cipher text : 129
Decrypted plain text : 5

Encryption and decryption using Lecture notes formula

Encrypted cipher text : 129
Decrypted plain text : 5
```