Network Security (CSE 350)

Assignment-2

Prof. B.N. Jain

Project 2 : AES Encryption/Decryption

Algorithm:
(1363+1362)%2=1

Submitted by:

Name	Roll No
Vansh	2021363
V. Bharath Krishna	2021362

Brief Overview of Project

Programming Language: C++
Platform: MS Windows

AES Specifications for our implementation:

Key size used: 128 bits/16 bytes

Total number of rounds: 10

4 Plaintexts taken

Plaintexts size: 128 bits/16 bytes

Helper functions:

copy_data(): it copies data of 4x4 matrix of strings from matrix 1 to matrix 2

hex_string_to_binary(): This function creates a binary string from a hexadecimal

string.binary_to_hex(): This function produces a hexadecimal string from a binary

string.xor_operation(): XORs four binary strings together.Galois Field (GF) multiplication is implemented by GF_multiplication() and is utilized in the AES MixColumns procedure.

matrix_to_string(): Generates a string from a 4x4 matrix of hexadecimal strings.

string_to_matrix(): Generates a 4x4 matrix of hexadecimal strings from a string.

GF (Galois Field) Multiplication:

- Galois Field (GF) multiplication is carried out on two hexadecimal strings that represent integers by this function, GF_multiplication. In the MixColumns stage of the AES (Advanced Encryption Standard) algorithm, which provides dispersion during the encryption process.
- it multiplies two hexadecimal values by GF by first converting them to binary, then carrying out the multiplication in binary, decreasing the result within the Galois Field, and then transforming the result back to hexadecimal before returning it.

Methods used:

- symmetric encryption technique in 128 bits of fixed-size blocks used to encrypt data by taking the input plain text of 128 bits/16 bytes. Original text in form of characters are converted into ASCII and this ASCII value is stored in the form of hexadecimals. 1 hexadecimal can represent 4 bit.
- the encryption process uses 10 rounds of operations, where each round consist of four steps - SubBytes, ShiftRows, MixColumns, and AddRoundKey
- Construction of Sbox and inverseSbox for substitution of bytes for encryption and decryption
- Taking the plaintext as input and performing the AES Encryption algorithm where the following operations substitue_bytes(),shift_rows(), Mix_Column() add_round_key() finding returning the result matrix_to_string(ciphertext)
- Similar to the AES Encryption process we decrypt the ciphertext using the AES Decryption algorithm using the inverse operations Inverse_Mix_Column(), inverse_shift_rows(), inverse_sub_bytes(), add_round_key() again returning the result matrix_to_string(ciphertext)
- Finally checking the 1st Encryption and the 9th Decryption output both will give the same hexadecimal values and vice-versa for 4 pairs of (plain_text, cipher_text)

AES Encryption process

- To store the intermediate and final encryption results, initialize the ciphertext and temporary matrices. The result in form of hexadecimals is converted to characters by converting the hexadecimal value into integer and finding the corresponding character to this integer ASCII value.
- Create a matrix from the supplied plaintext by using the string_to_matrix function.Using the starting round key (sub_key[0]), perform the add_round_key operation with the initial round_key
- For rounds 1 through 9, iteratively apply the substitue_bytes, shift_rows, Mix_Column, and add_round_key procedures.Following the initial encryption round, the intermediate ciphertext is kept in first_encryption.
- Tenth and Final Round: Use the procedures shift_rows, add_round_key, and substitute_bytes for this round.We get final ciphertext.

AES Decryption process

- Initialisation of the temp and plaintext matrices to hold the intermediate and final decryption results.
- Create a matrix from the provided ciphertext by utilizing the string_to_matrix function.The First Round:Utilizing sub_key[10] as the last round key, do the add_round_key function.
- Apply the procedures inverse_sub_bytes and inverse_shift_rows.Key Rounds 1 through 9:Iteratively apply the add_round_key, inverse_shift_rows, inverse_sub_bytes, and Inverse_Mix_Column procedures for rounds 9 to 1.
- the Final Round(or Zeroth round) :Utilizing sub_key[0] as the first round key, apply the add_round_key function.Once received, the final plaintext is Obtained.

Sample input/output:

```
PS C:\Users\vansh\Desktop\NSC-A2> cd "c:\Users\vansh\Desktop\NSC-A2\" ; if ($?) { g++ code.cpp -o code } ; if ($?) { .\code }
Key(128 bit) in hexadecimal: 5468617473206D79204B756E67204675

First Pair:
57686174736170702A596F7574756265, 3052D8148F3D687F773A1172E1C64BD6

Orignal text(in characters): Whatsapp*Youtube
Plaintext(in hexadeimals): 57686174736170702A596F7574756265
Ciphertext(in hexadecimals): 3052D8148F3D687F773A1172E1C64BD6
Cihertext(in characters): 0R╪¶Å=hw:◄rß╞K╔
Decypted Text(in characters): Whatsapp*Youtube
Decrypted Text(in hexadecimals): 57686174736170702A596F7574756265
First Encryption(in hexadecimals): E2C053F5DCE98E8F480841B7FC50ABE6
Ninth Decryption(in hexadecimals): E2C053F5DCE98E8F480841B7FC50ABE6
Ninth Encryption(in hexadecimals): 34F0D2B61E25D2DEB7F99B178EE96F4E
First Decryption(in hexadecimals): 34F0D2B61E25D2DEB7F99B178EE96F4E

Second Pair:
6D796D61726B73696E4E534331303078, 8BAD2050CC68F0191D0FF4583C527633

Orignal text(in characters): mymarksinNSC100x
Plaintext(in hexadecimals): 6D796D61726B73696E4E534331303078
Ciphertext(in hexadecimals): 8BAD2050CC68F0191D0FF4583C527633
Cihertext(in characters): ï¡ P╠h≡↓⇔⌈X<Rv3
Decrypted Text(in hexadecimals): 6D796D61726B73696E4E534331303078
Decypted Text(in characters): mymarksinNSC100x
First Encryption(in hexadecimals): 282B8711307BDC2485943996E84A5C97
Ninth Decryption(in hexadecimals): 282B8711307BDC2485943996E84A5C97
Ninth Encryption(in hexadecimals): 71346CEE2E1F97B23327832F880B0805
First Decryption(in hexadecimals): 71346CEE2E1F97B23327832F880B0805

Thrid Pair:
656B6B686F6B61646F40646976696E65, 5B73306A1E3CE98DCCEFA557FFAE77A7

Orignal text(in characters): ekkhokado@divine
Plaintext(in hexadecimals): 656B6B686F6B61646F40646976696E65
Ciphertext(in hexadecimals): 5B73306A1E3CE98DCCEFA557FFAE77A7
Cihertext(in characters): [s0j▲<0i╠nÑW «wº
Decrypted Text(in hexadecimals): 656B6B686F6B61646F40646976696E65
Decypted Text(in characters): ekkhokado@divine
First Encryption(in hexadecimals): F17C9F3AC5AF187A5C60D50B039FD268
Ninth Decryption(in hexadecimals): F17C9F3AC5AF187A5C60D50B039FD268
Ninth Encryption(in hexadecimals): 8FAE09784DE9F27639E2559138594C70
First Decryption(in hexadecimals): 8FAE09784DE9F27639E2559138594C70
```