Title of the Experiment: Information Security: Encryption/Decryption

Group Members' Names: Sujan Kumar Saha, Pankaj Bhowmik

Date: 9/03/2019

Abstract

The objective of this experiment is to learn about encryption and decryption algorithms, how these algorithms work and how to use the encryption and decryption algorithms to secure the confidential information. In this experiment, we worked with two algorithms: Caesar Cipher and Advanced Encryption Standard (AES). In the first part, Caesar Cipher is implemented using C code and its functionality is tested. The code can encrypt data provided in the form of English capital Alphabets and decrypting the data successfully when provided with the key. In the second part of the experiment, a sample C code for AES encryption is provided. The sample code is capable of encrypting and decrypting 128 bits of data. As per the instructions provided in the lab manual, several modifications are done to better understand the possible limitations of the implementation. The provided code was incapable to analyze any length of data. As the final task of the experiment, the provided C code is modified accordingly to enhance its capability up to encrypting and decrypting with key of 192 bits and 256 bits. While modifying the sample code for 192 and 256 bits, the number of rounds in key rounding is changed.

Experimental Results

Goals: The goal of this experiment is to understand how encryption algorithms work and how to use these algorithms to encrypt and decrypt the confidential data.

Experimental Setup: We used Linux operating system (CentOS 7) on virtual box on our own computers. We compiled those C codes using gcc compiler and used 32 bit systems.

Part 1:
Experiment Steps:

1. Caesar Cipher: The C code exp2.c has been compiled with gcc compiler with the command gcc exp2.c -o exp2. The code has been run with the key input range -26 to 26. The plain text should have all English capital letters. The output has been observed.
2. Advanced Encryption Standard (AES): The given codes have been compiled and observed the output according to the given instruction.

Results and Observations

1. Caeser Cipher : The output screen shot is given below.

```
[ssaha@localhost Exp2]$ ./exp2
Positive interger makes the left shift and negative integer will make right shift.
The key value has the range -26 to 26.
Please enter the key:
5
The plain text should contain only capiital letters
Please give the plain text
PANKAJ
The cypher text:
UGTQGP
[ssaha@localhost Exp2]$
```

Figure 1: The Ceasar Cipher output

2. AES: The screenshot for AES step 5 is given below.

```
-rwxrwxr-x. 1 ssaha ssaha 21160 Aug 28 00:48 a.out
-rw-r--r--. 1 ssaha ssaha   961 Aug 29  2016 test.c
-rw-r--r--. 1 ssaha ssaha 17866 Aug 29  2016 TI_aes.c
-rw-r--r--. 1 ssaha ssaha   164 Aug 29  2016 TI_aes.h
[ssaha@localhost AEScode]$ ./a.out
Plain Text Before Encryption TXT  : 'gainesvillekjhkjhlkjlkjnkjnkjnlklkmlkmn'
Plain Text Before Encryption HEX  : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x6b 0x6a 0x68 0x6b 0
x6a
Cipher Text After Encryption      : '��z�k�[�(A'
Cipher Text After Encryption HEX  : 0x83 0xf9 0x19 0x2d 0x7a 0xcc 0x6b 0xee 0x5b 0x85 0x28 0x16 0x1a 0xe 0xa1 0x
41
Plain Text After Decryption       : 'gainesvillekjhkj'
Plain Text After Decryption HEX   : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x6b 0x6a 0x68 0x6b 0
x6a
```

Figure 2: AES 128 bit output with a long input

The plaintext and cypher texts are totally different here. The cypher texts can not be read here. But it's length is 16 character whereas the plaintext is very long.

3. AES: The screenshot for AES step 8 is given below.

```
[ssaha@localhost AEScode]$ gcc -w test.c TI_aes.c
[ssaha@localhost AEScode]$ ./a.out
Plain Text Before Encryption TXT  : 'gainesville'
Plain Text Before Encryption HEX  : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x0 0x0 0x0 0x0 0x0
Cipher Text After Encryption      : 'n�L_�2'�e�C+�~'
Cipher Text After Encryption HEX  : 0x1a 0x1e 0x6d 0x9d 0x4c 0x5f 0xf7 0x32 0x27 0x84 0x65 0xbd 0x43 0x2b 0xa2 0
x7e
Plain Text After Decryption       : 'gainesville'
Plain Text After Decryption HEX   : 0x67 0x61 0x69 0x6e 0x65 0x73 0x76 0x69 0x6c 0x6c 0x65 0x0 0x0 0x0 0x0 0x0
[ssaha@localhost AEScode]$
```

Figure 3: AES 128 bit output with an input less than 16 characters

Here, the hex values have been changed according to the AES encryption algorithm for all the characters in the plain text.

4. AES: The screenshot for AES step 10 is given below:

```
[ssaha@localhost AEScode]$ gcc -w test.c TI_aes.c
[ssaha@localhost AEScode]$ ./a.out
Plain Text Before Encryption TXT  : 'universityoffloridauniversityofflorida'
Plain Text Before Encryption HEX  : 0x75 0x6e 0x69 0x76 0x65 0x72 0x73 0x69 0x74 0x79 0x6f 0x66 0x66 0x6c 0x6f 0
x72
Cipher Text After Encryption      : 'J��4;%��a�l'
Cipher Text After Encryption HEX  : 0x4a 0x7 0x9e 0x8b 0x34 0x3b 0xc1 0x25 0xcb 0xe3 0x61 0xa3 0x6c 0x1a 0x14 0x
c4
Plain Text After Decryption       : 'universityofflor'
Plain Text After Decryption HEX   : 0x75 0x6e 0x69 0x76 0x65 0x72 0x73 0x69 0x74 0x79 0x6f 0x66 0x66 0x6c 0x6f 0
x72
[ssaha@localhost AEScode]$
```

Figure 4: AES 128 bit output with an input greater than 16 characters

(a) The text matches upto 16 character length as the AES work for 128 bit.

(b) Yes, the key length can be changed to 256 bits. After changing the key length to 256 bit and also by changing the number of rounds, we can encrypt and decrypt long plain texts up to 256 bits.

(c) Advantages and disadvantages of 192 and 256 bit key lengths: The main advantage of a longer key like 192 bit or 256 bit is its resiliency to brute force attack. When AES was first introduced, breaking 128-bit key was infeasible for the resources available then. But with present processing ability, attackers could break 128bit AES, and even 192bit AES. 256 bit AES has yet not been broken. The longer the key ensures a higher level of security in the encryption. The only disadvantage with longer key AES is its requirement for memory and time. During processing, it takes up more memory space and extra rounds take extra time. However, given the processing power, speed and memory availability of present day, this problem does not cause a big problem and can be overlooked.

Summary:

The purpose of the experiment was to help learners get acquainted with standard encryption algorithms, help students understand how these work, and guide them in the implementation of two encryption algorithms. In pursuit of the objectives, we wrote a C code for the implementation of Caesar Cipher and verified its functionality. Afterwards, we were given a sample C code for AES encryption of 128 bits. To help us better understand the mechanism of AES encryption, we were asked to try several modifications and explain the changes caused by them.