CNT 5410 Computer Network Security
Assignment 1

Mugdha Kumar (UFID: 54219168)

January 29, 2015

Introduction In this assignment we are doing Encryption/Decryption/Hashing using the *gcrypt* libraries provided by the Linux operating system . We are providing the input file through command prompt. Results are getting stored in **finalRecordFile.txt** file in the same folder. Once it accepted the input file. I will calculate the Mean and Median time (Milli Seconds) taken by each of the below mentioned algorithms

- AES 128 CBC Encrytion And Decryption
- AES 256 CBC Encrytion And Decryption
- RSA 1024
- RSA 4096
- HMAC MD5
- HMAC SHA1
- HMAC SHA256
- Digital Signature using HMAC 256 and RSA 4096. . .

AES 128 CBC Encrytion And Decryption In AES 128 CBC, cipher is *GCRY CIPHER AES128* and cipher mode is *GCRY CIPHER MODE CBC*. First a key of length 16 bytes is generated and set to the handle. Same is done for 'Initialization Vector[IV]'.After that plain text is encrypted and decrypted in sequence using the same key as it's a symmetric algorithm. This process is repeated 100 times. After that Mean time and Median time is calculated and stored in "finalRecordFile.txt" file. Encrypted and decrepted contents of input files are stored at **AES128 Encrpy.txt** and **AES128 Decrpy.tx** respectively. Below Tested on .txt,.mp4,.png and .pdf files. Maximum file size on which testing is done is 407MBsTime is in **Milli Seconds.**

File Name	Size	EncrMean time	EncrMedian time	DecrMean time	DencrMedian time
test.txt	100Bytes	0.0090	0.0090	0.0160	0.0160
BFile.txt	2.6MBs	43.1227	42.92550	48.6974	48.59950
picture.png	320KBs	5.2553	5.1885	5.5333	5.5055
test.mp4	354MBs	5808.7578	5808.7578	6192.7109	6192.7109
test1.mp4	407MBs	6947.0592	6947.0592	7122.3418	7122.3418
cpp.pdf	2.6MBs	43.0555	42.9695	45.6159	45.5850

AES 256 CBC Encrytion And Decryption Repeated the same process which I did in AES 128 CBC, just change the key length to 32 Bytes and changed cipher to ' *AES 256 CBC*'.Here also each encrytion and decrytion is run 100 times and after that Mean time and Median time is calculated and stored in "finalRecordFile.txt" file. Encrypted and decrepted contents of input files are stored at **AES256 Encrpy.txt** and **AES256 Decrpy.tx** respectively. Below are some report data on some sample input files. Tested on .txt,.mp4,.png and .pdf files. Maximum file size on which testing is done is 407MBsTime is in **Milli Seconds**

File Name	Size	EncrMean time	EncrMedian time	DecrMean time	DencrMedian time
test.txt	100Bytes	0.0100	0.0100	0.0170	0.0170
BFile.txt	2.6MBs	53.7804	53.6350	62.4599	62.3865
picture.png	320KBs	6.7231	6.6960	7.1278	7.1150
test.mp4	354MBs	7792.6988	7792.6988	8108.9198	8108.9198
test1.mp4	407MBs	8535.2867	8535.2867	9238.2822	9238.2822
cpp.pdf	2.6MBs	56.8290	56.8290	58.0730	58.0730

RSA 1024 It is an example of asymmetric algorithm in which public and private keys are different. Encryption is done using public key and decryption is done using private key. This algorithm could run 37 times only out of 100 times on file of 100Bytes. After that it is throwing error *"Fatal error: out of core in secure memory "*. I tried to debug it a lot but could not find the exact reason. Mean and Median is calculated for the iteration it went through and stored in "finalRecordFile.txt" file. Encrypted and decrypted contents of input files are stored at **RSA1024 Encrpy.txt** and **RSA1024 Decrpy.txt** respectively.For running RSA 1024, we need to comment all other algorithm function call and run only rsa 1024. Also need to change the ITEARTION count to 35 for file size 100Bytes. For running file size of 3.1KBs, change the ITERATION Count to 1 only. Maxmimum file size on it works is 3.3KBs. I have defined a variable ITERATION in main file for counting the number of round it Time is in **Milli Seconds**

File Name	Size	EncrMean time	EncrMedian time	DecrMean time	DencrMedian time
test.txt	100Bytes	1.2089	1.2105	41.1312	39.9185
test1.txt	3.1KBs	33.0410	33.0410	1285.8530	1285.8530

RSA 4096 For RSA 4096 whole program is implemented, just unable to resolve error B*ad character in S-expression*. I have code for encryption and decryption which is similar to RSA 1024.

HMAC MD5 Hash algorith used for HMAC MD5 is *GCRY MD MD5*. First we created a message digest object for the algorithm . After that create a key of length 64Bytes. There is no restriction on key length. I have used 64Byte here.After hashing is done digest object would be updated with digest values by handles. tested on various file types like .txt, .png, .pdf and .mp4. Below are their mean and median time. Also program will store the hashed data in **HMAC MD5 HashValue.txt** file Time is in **Milli Seconds**

File Name	Size	Hash Mean time	Hash Median time
test.txt	100Bytes	0.0110	0.0110
BFile.txt	2.6MBs	9.5770	9.5770
picture.png	320KBs	1.8390	1.8390
test.mp4	354MBs	1145.4771	1145.4771
test1.mp4	407MBs	1369.8450	1369.8450
cpp.pdf	2.6MBs	9.5260	9.5260

HMAC SHA1 We did same as HMAC MD5 in HMAC SHA1 also, only changes the algorithm to *GCRY MD SHA1*.Tested on various file types like .txt, .png, .pdf and .mp4. Below are their mean and median time. Also program will store the hashed data in **HMAC SHA1 HashValue.txt** file. Time is in **Milli Seconds**

File Name	Size	Hash Mean time	Hash Median time
test.txt	100Bytes	0.0147	0.0100
BFile.txt	2.6MBs	20.1216	19.6900
picture.png	320KBs	2.4346	2.3835
test.mp4	354MBs	2722.9851	2722.9851
test1.mp4	407MBs	3042.7100	3042.7100
cpp.pdf	2.6MBs	19.9777	19.9585

HMAC SHA256 We did same as HMAC MD5 in HMAC SHA256 also, only changes the algorithm to *GCRY MD SHA256*.Tested on various file types like .txt, .png, .pdf and .mp4. Below are their mean and median time. Also program will store the hashed data in **HMAC SHA256 HashValue.txt** file. Time is in **Milli Seconds**

File Name	Size	Hash Mean time	Hash Median time
test.txt	100Bytes	0.0178	0.0120
BFile.txt	2.6MBs	34.7741	34.7170
picture.png	320KBs	4.4201	4.3570
test.mp4	354MBs	4828.6040	4828.6040
test1.mp4	407MBs	5631.8940	5631.8940
cpp.pdf	2.6MBs	36.1720	36.1720

Digital Signature using HMAC SHA 256 and RSA 4096

For calculating the digital signature, first we took the large input file, hashed it using HMAC SHA 256. After that generated public key and private key of 4096 bits. For signing, encrpyted the hashed file using private key. Also verified using private key if signature is good or not. Below is time take for siging the hashed file in **Milli second**

File Name	Size	Time taken
test.txt	100Bytes	0.0150
BFile.txt	2.6MBs	0.0160
picture.png	320KBs	0.0140
test.mp4	354MBs	0.0160
test1.mp4	407MBs	0.0160
cpp.pdf	2.6MBs	0.0150

textbfResult On console, I am printing the starting and ending of any algorithm. Mean and Median time are stored in file. Below is the screenshot of console.

```
mugdha@mugdha-HP-Pavilion-dv6-Notebook-PC:~/Documents/Assignment2CNT5410/gatorCrypto$ ./a.out test.txt
InputFileName: test.txt
--aes 128 CBC Encryption and Decryption started---
---aes 128 CBC Encryption and Decryption Completed----
--aes 256 CBC Encryption and Decryption started--
aes 256 CBC Encryption and Decryption Completed
HMAC_SHA1 Hash function started
HMAC_SHA1 Hash function Completed
HMAC_SHA_MD5 Hash function started
HMAC_SHA_MD5 Hash function Completed
HMAC_SHA256 Hash function started
HMAC_SHA256 Hash function Completed
HMAC_SHA256 _RSA 4096 Signature started
HMAC_SHA256 _RSA 4096 Signature Completed

finalRecordFile.txt file is recording all the time taken. Please check it for result
```

```
mugdha@mugdha-HP-Pavilion-dv6-Notebook-PC:~/Documents/Assignment2CNT5410/gatorCrypto$ ./a.out test.txt
InputFileName: test.txt
-----RSA 1024 started----------
-----RSA 1024 Completed----------

finalRecordFile.txt file is recording all the time taken. Please check it for result
```

Conclusion Based on the data collected for various algorithm, below is their order(Highest to lowest performance)

1. Digital signature generation using HMAC 256 and RSA 4096
2. HMAC MD5
3. HMAC SHA1
4. AES128, CBC Mode
5. AES256, CBC Mode
6. RSA1024
7. RSA4096 . . .

System Specification All testing is done on machine whose specification is below:
Model Name Intel(R) Core(TM)2 Duo CPU T6400 @ 2.00GHz
address sizes : 36 bits physical, 48 bits virtual
cpu MHz : 1200.000
MemTotal: 4097368 kB