**CSE 410/565 SPRING 2021**

**HOMEWORK 2 - REPORT**

**SURYA MUTHIAH PILLAI (UB person no.: 50363234)**

**Email:** [**suryamut@buffalo.edu**](mailto:suryamut@buffalo.edu)

**Objective:**

To implement the given tasks in the Homework using cryptographic functions and measure the time taken for each of those tasks based on the requirement.

**Implementation:**

**Language Used:** Python3 (3.8.5)

**Platform:** Windows 10 64-bit/ Virtual Machine (Ubuntu-20.04.1 x86-64)

**Cryptographic Package:** Pycryptodome 3.10.1

**Files submitted:** cryptotools.py, Plain\_Text\_1KB, Plain\_Text\_10MB, Plain\_Text\_1MB (for RSA)

**Running the code in terminal:**

*Package installation:*

The additionally required packages can be installed with the following commands

*sudo apt install python3-pip* (pip3 installation)

*sudo apt-get install build-essential python3-dev (for pycryptodome)*

*pip3 install pycryptodome* (pycryptodome install)

Python is used to code the encryption/decryption system which is submitted in the compressed zip file as ‘cryptotools.py’. Once the contents of the zip file are extracted into a directory, simply redirect into the directory of the python file using cd command and type the following commands to run the non-interactive code. (Please make sure all the above-mentioned Files are present in the same directory)





**Performance measurement:**

There are a total of 8 tasks given, (a) – (h) on which we are going to perform encryption, decryption, hashing etc. on the tasks and measure the time taken for comparison based on the requirement for each of the corresponding tasks in order to produce a report which can give an idea about the performance of the different encryption, hashing and other mechanisms we have implemented in the above python file named ‘cryptotools.py’.

***Note:*** Before we move on to the performance check, one more thing to note is RSA cannot process the entire data at once on either 1KB or 1MB file. So as discussed in the lecture, we split the data into chunks of certain size for the RSA algorithm in pycrypto to process our data.

For 2048-bit key RSA in OAEP, we use

chunk\_size = (2048/8) – 42 = 214 bytes/chunk

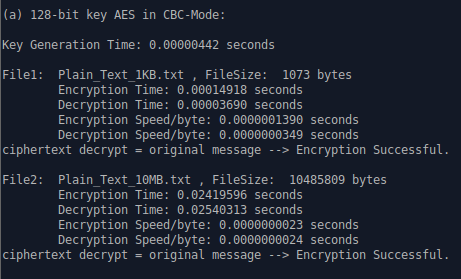
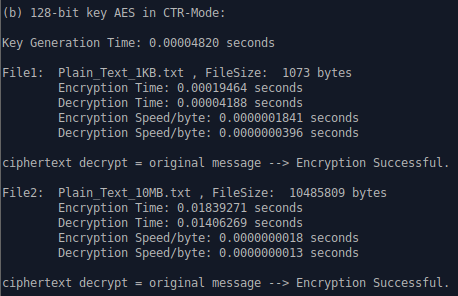
For 3072-bit key RSA in OAEP, we use

chunk\_size = (3072/8) – 42 = 342 bytes/chunk

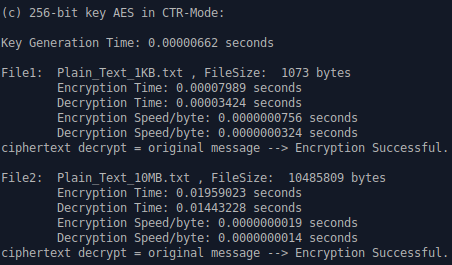
(Chunk size and number of chunks split up for both 1KB and 1MB file are mentioned in the RSA results below as well)

First, let’s look at the output of different algorithms to evaluate our results. (After running the code a few times, the time measured for the different algorithms averaged around the values as following)

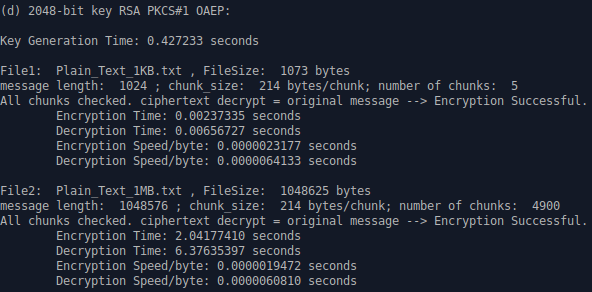
1. 128-bit key AES (CBC) (b) 128-bit key AES (CTR)

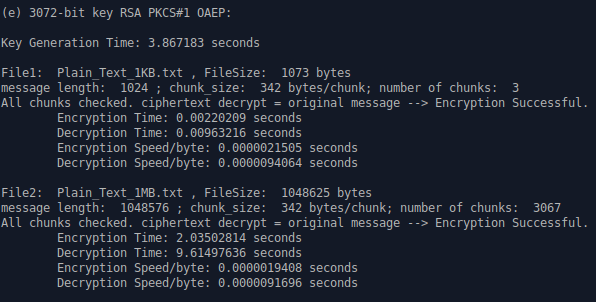
(c) 256-bit key AES (CTR)



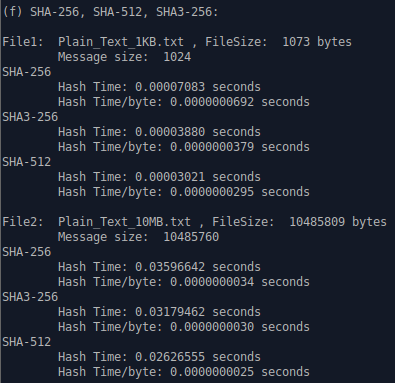
(d) 2048-bit key RSA PKCS#1 OAEP:



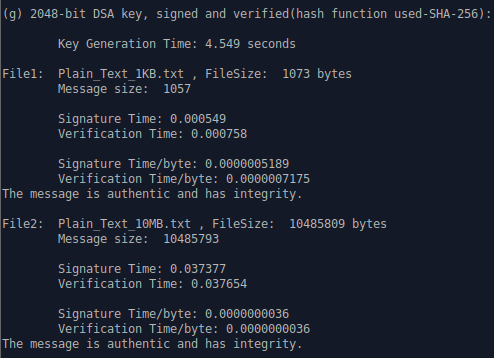
(e) 3072-bit key RSA PKCS#1 OAEP:



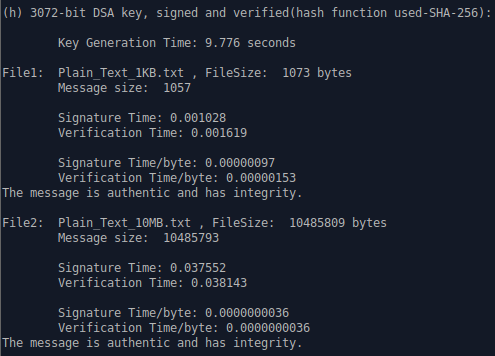
(f) SHA-256, SHA-512, SHA3-256:



(g) 2048-bit DSA key, signed and verified (hash function used-SHA-256)



(h) 3072-bit DSA key, signed and verified (hash function used-SHA-256)



1. **how per byte speed changes for different algorithms between small and large files:**

(a), (b), (c) – AES Encryption/Decryption:

Irrespective of the key-size and the mode used in tasks (a)-(c) where we use AES Encryption in CBC and CTR mode (128- & 256-bit keys in CTR), the difference in per byte speed change between the 1KB and 10MB file remains the same i.e., The larger file’s (10MB) Encryption/byte speed is significantly faster (atleast by a factor of 40) than the smaller file’s (1KB) Encryption/byte speed.

In the case of decryption, the same trend as above follows i.e., The Decryption-speed/byte is higher for the larger file (10MB) but with the only difference that the speed is higher than for the smaller file (1KB) by a factor of atmost 30.

(d), (e) – RSA PKCS#1 OAEP Encryption/Decryption:

In both RSA implementations with 2048- and 3072-bit key, the per byte speed for Encryption for both files irrespective of their sizes are almost the same with negligible difference of around 4 nanoseconds.

The same goes for Decryption too as the per byte speed for decryption is also almost similar for both the files with the same negligible difference of about 3-4 nanoseconds.

(f) – HASHING - SHA-256/512, SHA3-256:

The trend is similar to that of AES Encryption. The Hashing Speed per byte for the larger file of 10MB is faster than for the smaller file (1KB) by a factor of 10 to 20.

At certain run-times (very rarely), the Hashing speed per byte for the smaller file is slightly even slower than normal, in which case the factor might go up to 30.

(g), (h) – DSA:

The same goes for DSA as well. The Signing/Verifying Speed per byte for the large file is significantly faster by atleast 150 times than that for the small file. This applies to both the implementation of DSA with 2048-bit key and 3072-bit key.

1. **how encryption and decryption times differ for a given encryption algorithm:**

The speed or time comparison between the small and large files is similar in the case of encryption or decryption of any algorithm. The time taken to encrypt/decrypt any file for any given algorithm at any given point is directly proportional to that of the file size. Larger the file size -> more the time taken to encrypt/decrypt the file. This goes for all the algorithms implemented including the hashing and signing mechanisms.

Comparison b/w Encryption and Decryption:

(a), (b), (c) – AES:

Irrespective of the key size used, Encryption in AES (both CTR and CBC mode) is slower than decryption by a small factor for any given file.

(e), (f) – RSA:

On the other hand, Encryption is faster than Decryption in the case of RSA (when both 2048/3072-bit keys are used) by a very small factor.

1. **how key generation, encryption, and decryption times differ with the increase in the key size:**

(a), (b), (c) – AES:

128- and 256-bit key generation time is almost the same.

In AES-CTR mode, when the key size is increased from 128-bits to 256-bits, the encryption and decryption time is almost same for both the cases for any given file.

Hence, there are no significant changes in time when key-size is increased for a given file.

(e), (f) – RSA:

2048-bit key takes less than 1 second to be generated while 3072-bit key (which is 210 times bigger than that of the 2048-bit key) takes atleast an average of 3.5 seconds to be generated.

Despite the fact that 3072-bit key is 1000 times bigger than 2048-bit key, it takes only a few times more than the key generation time of a 2048-bit RSA key which is highly advantageous.

Even though the key generation time differs, irrespective of the key size, be it 2048-bits or 3072-bits, the encryption/decryption time for any file for RSA algorithm is almost the same.

Therefore, increase in key-size only increases the key generation time and not the encryption/decryption time for a given file.

1. **how hashing time differs between the algorithms and with increase of the hash size:**

SHA-256/512:

When hash size is increased from 256-bits to 512-bits, the hashing time is faster. So, larger the hash size, faster the hashing is done.

SHA3-256 vs SHA-256/512:

Comparing SHA and SHA3, SHA3-256 performs better than SHA-256 but not as good as SHA-512.

So, performance-wise we can order SHA-512 to hash the fastest and then the SHA3 algorithm and at last the SHA-256 which is the slowest among the 3 hashing mechanism we have used.

1. **how performance of symmetric key encryption (AES), hash functions, and public-key encryption (RSA) compare to each other:**

AES (symmetric key encryption) which uses a single key for both encryption and decryption is significantly faster than RSA (public-key encryption) which uses two separate keys (public and private key) for encryption/decryption especially in the case of large files.

Therefore, AES Encryption is faster than RSA in terms of performance but less secure than RSA.

However, when compared to hashing time, AES almost performs similar to the hashing algorithms i.e., Encryption time in AES and Hashing time using SHA is almost the same. While on the other hand, RSA is slower than both AES and hashing algorithms used.