# CSE 410/565 Spring 2023 - Homework 2

Darryl Vas Prabhu

**Performance report**

I. The chosen programming language - **Python3** Library chosen - pycryptodomex

II. The timing results that your program measured as specified above:
Ans: Ouput parameters: All time related paramters is given in nanoseconds (ns)
**Filename**: Name-of-the-file:size-of-the-file
**KeyGen Time (ns)** : The time it takes to generate a new key.
**Enc Time (ns)**: The total time it takes to encrypt each file
**Dec Time (ns)**: The total time it takes to decrypt each file
**Encryption speed per byte(ns)**= size-in-bytes/Enc Time: encryption speed per byte for the file
**Decryption speed per byte(ns)**=size-in-bytes/Dec Time : decryption speed per byte for the file
**Hash Time (ns)**: The total time to compute the hash of the file
**Per-Byte-Hash (ns)**: per-byte hash time.
**Sign Time (ns)**: Time to produce a signature for the file
**Verify Time (ns)**: Time to verify the signature for the file
**Per Byte Sign time (ns)**: signature per byte for the file
**Per Byte Verify time (ns)**: Verification per byte for the file

Timing results:

```
----------------------AES CTR encryption/decryption----------------------
--------------------
Original message slice:  b'What is Lo'
The received message is : b'What is Lo'

SUCCESS
AES-CTR 128-bit key with 1KB file :
 Filename: package_1:1KB ,
 KeyGen Time :2858 ns ,
 Enc Time: 49984 ns,
 Dec Time: 10475 ns ,
 Encryption speed per byte:49.984 ns,
 Decryption speed per byte:10.475 ns
--------------------


Original message slice:  b'FdciWQzrhW'
The received message is : b'FdciWQzrhW'

SUCCESS
AES-CTR 128-bit key with 10MB file :
 Filename: package_2:10MB ,
 KeyGen Time :4611 ns ,
 Enc Time: 23119086 ns,
 Dec Time: 18302604 ns ,
 Encryption speed per byte:2.2048078536987306 ns,
 Decryption speed per byte:1.7454723358154296 ns
--------------------
```

```
Original message slice:  b'What is Lo'
The received message is : b'What is Lo'

SUCCESS
AES-CTR 256-bit key with 1KB file :
 Filename: package_1:1KB ,
 KeyGen Time :3904 ns ,
 Enc Time: 20510 ns,
 Dec Time: 8462 ns ,
 Encryption speed per byte:20.51 ns,
 Decryption speed per byte:8.462 ns
--------------------

Original message slice:  b'FdciWQzrhW'
The received message is : b'FdciWQzrhW'

SUCCESS
AES-CTR 256-bit key with 10MB file :
 Filename: package_2:10MB ,
 KeyGen Time :5493 ns ,
 Enc Time: 27487629 ns,
 Dec Time: 19919109 ns ,
 Encryption speed per byte:2.6214245796203612 ns,
 Decryption speed per byte:1.8996342658996581 ns

----------------------AES CBC encryption/decryption----------------------
--------------------


Original message slice:  b'What is Lo'
The message was  b'What is Lo'


SUCCESS
AES-CBC 128-bit key with 1KB file :
 Filename: package_1:1KB
 KeyGen Time :4302 ns ,
 Enc Time: 67625 ns,
 Dec Time: 11306 ns ,
 Encryption speed per byte:67.625 ns,
 Decryption speed per byte:11.306 ns
--------------------


Original message slice:  b'FdciWQzrhW'
The message was  b'FdciWQzrhW'


SUCCESS
AES-CBC 128-bit key with 10MB file :
 Filename: package_2:10MB
 KeyGen Time :4932 ns ,
 Enc Time: 39320558 ns,
 Dec Time: 21372078 ns ,
 Encryption speed per byte:3.7499006271362303 ns,
 Decryption speed per byte:2.0382001876831053 ns
--------------------
```

```
Original message slice:  b'What is Lo'
The message was  b'What is Lo'


SUCCESS
AES-CBC 256-bit key 1KB file :
 Filename: package_1:1KB
 KeyGen Time :4931 ns ,
 Enc Time: 31191 ns,
 Dec Time: 10161 ns ,
 Encryption speed per byte:31.191 ns,
 Decryption speed per byte:10.161 ns
--------------------


Original message slice:  b'FdciWQzrhW'
The message was  b'FdciWQzrhW'


SUCCESS
AES-CBC 256-bit key with 10MB file :
 Filename: package_2:10MB
 KeyGen Time :6181 ns ,
 Enc Time: 34972973 ns,
 Dec Time: 18740417 ns ,
 Encryption speed per byte:3.3352826118469237 ns,
 Decryption speed per byte:1.7872254371643066 ns

-----------------------HASHING-----------------------

SHA-256 with 1KB file :
 Hashing function : SHA-256
 Filename: package_1:1KB
 Hash Time :14112 ns ,
 Per-Byte-Hash : 14.112 ns


SHA-256 with 10MB file :
 Hashing function : SHA-256
 Filename: package_2:10MB
 Hash Time :52893223 ns ,
 Per-Byte-Hash : 5.044290828704834 ns


SHA-512 with 1KB file :
 Hashing function :SHA-512
 Filename: package_1:1KB
 Hash Time :14403 ns ,
 Per-Byte-Hash : 14.403 ns


SHA-512 with 10MB file :
 Hashing function : SHA-512
 Filename: package_2:10MB
 Hash Time :44840316 ns ,
 Per-Byte-Hash : 4.276305770874023 ns


SHA3-256 with 1KB file :
```

```
 Hashing function : SHA-3-256
 Filename: package_1:1KB
 Hash Time :23122 ns ,
 Per-Byte-Hash : 23.122 ns


SHA3-256 with 10MB file :
 Hashing function : SHA-3-256
 Filename: package_2:10MB
 Hash Time :42274589 ns ,
 Per-Byte-Hash : 4.031618976593018 ns

-----------------------RSA encryption/decryption-----------------------


SUCCESS
2048 bit key with 1KB file :
Filename: package_1:1KB
KeyGen Time :474146880 ns,
Enc Time: 5971043 ns,
Dec Time: 14925220 ns ,
Encryption speed per byte:5971.043ns,
Decryption speed per byte:14925.22ns


SUCCESS
2048 bit key with 1MB file :
 Filename: file_rsa:1MB
 KeyGen Time :1297416992 ns,
 Enc Time: 3051867871 ns,
 Dec Time: 12597736426 ns ,
 Encryption speed per byte:2910.4880056381226ns,
 Decryption speed per byte:12014.137674331665ns


SUCCESS
3072 bit key with 1KB file :
 Filename: package_1:1KB
 KeyGen Time :742012294 ns,
 Enc Time: 12678397 ns,
 Dec Time: 35097378 ns ,
 Encryption speed per byte:12678.397ns,
 Decryption speed per byte:35097.378ns


SUCCESS
3072 bit key with 1MB file :
 Filename: file_rsa:1MB
 KeyGen Time :844777536 ns,
 Enc Time: 6317607041 ns,
 Dec Time: 30990136737 ns ,
 Encryption speed per byte:6024.939576148987ns,
 Decryption speed per byte:29554.49746799469ns

-----------------------DSA-----------------------


The message is authentic
2048 bit key with 1KB file :
```

```
Filename: package_1:1KB  ,
Keysize:2048 bits,
KeyGen Time :10191672669 ns,
Sign Time: 931365 ns,
Verify Time: 1432066 ns,
Per Byte Sign time: 931.365 ns,
Per Byte Verify time: 1432.066 ns


The message is authentic
2048 bit key with 10MB file :
Filename: package_2:10MB  ,
Keysize:2048 bits,
KeyGen Time :216168680 ns,
Sign Time: 889261 ns,
Verify Time: 1073429 ns,
Per Byte Sign time: 0.08480653762817383 ns,
Per Byte Verify time: 0.10237016677856445 ns


The message is authentic
3072 bit key with 1KB file :
Filename: package_1:1KB  ,
Keysize:3072 bits,
KeyGen Time :46853992024 ns,
Sign Time: 5788817 ns,
Verify Time: 1766097 ns,
Per Byte Sign time: 5788.817 ns,
Per Byte Verify time: 1766.097 ns


The message is authentic
3072 bit key with 10MB file :
Filename: package_2:10MB  ,
Keysize:3072 bits,
KeyGen Time :26867206311 ns,
Sign Time: 1791853 ns,
Verify Time: 2123707 ns,
Per Byte Sign time: 0.17088441848754882 ns,
Per Byte Verify time: 0.20253248214721 68 ns
```

III. For each performance aspect below, your comments about (i) the expected performance (ii) whether the observed performance followed the expected performance (iii) if there was a difference, your justification of the difference:

**1.**[2 points for 410/565] how per byte speed changes for different algorithms between small and large files.
**Ans** Comparison between AES-CTR and AES-CBC per byte speed for small (1KB) and large (10MB) files and Hash-512 and Hash3-256.

**Algorithm AES CTR 128-bit key:**
1KB file:
Encryption speed per byte: 49.984 ns,
Decryption speed per byte: 10.475 ns
10MB file :
Encryption speed per byte: 2.2048078536987306 ns,
Decryption speed per byte: 1.7454723358154296 ns

**Algorithm AES-CBC 128-bit key:**
1KB file:
Encryption speed per byte: 67.625 ns,
Decryption speed per byte: 11.306 ns
10MB file :
Encryption speed per byte: 3.7499006271362303 ns,
Decryption speed per byte: 2.0382001876831053 ns
Per byte encryption time for AES algorithm in both CBC and CTR mode decreases with increase in the size of the file. Encryption per byte and decryption per byte time almost approaches unit nanosecond in both cases when the size of the file increases by almost $10^4$ times.

**Algorithm Hash: SHA-256**
1KB file:
Per-Byte-Hash : 14.112 ns
10MB file :
Per-Byte-Hash : 5.044290828704834 ns

**Algorithm Hash: SHA3-256**
1KB file:
Per-Byte-Hash : 23.122 ns
10MB file :
Per-Byte-Hash : 4.031618976593018 ns
Per byte hash time for SHA3-256 decreases with increase in the file-size as observed from the experiment.

2.[2 points for 410/565] how encryption and decryption times differ for a given encryption algorithm;
**Ans** :
**Algorithm RSA:**
2048 bit key encryption/decryption with 1MB file
Filename: file_rsa: 1MB
Enc Time: 3051867871 ns,
Dec Time: 12597736426 ns
We can see that as expected the encryption time for RSA is less than the decryption time. This is because determining the decryption exponent value is much larger than the encryption exponent value. As expected decryption time is longer and almost 4x times the encryption time.

3.[2 points for 410/565] how key generation, encryption, and decryption times differ with the increase in the key size;
**Ans** : **Algorithm RSA: encryption/decryption**
2048 bit key with 1MB file :
KeyGen Time :1297416992 ns,
Enc Time: 3051867871 ns,
Dec Time: 12597736426 ns ,

3072 bit key with 1MB file :
KeyGen Time :844777536 ns,
Enc Time: 6317607041 ns,
Dec Time: 30990136737 ns

From the experiment , we observe that all 3 times, encryption decryption and key generation time ought to increase. We see that encryption and decryption time increases with the increase in the modulus, however there is decrease in the key-generation time. Suspect this to be due to already calculated values 3072 for the previous 1KB file, however more needs to be checked on why this value is decreased with increase in key size.

4.[2 points for 410/565] how hashing time differs between the algorithms and with increase of the hash size;

**Ans** : Hashing time increases with the increase in the hash size. SHA3-256 has the highest hashing time( uses sponge functions), with SHA-512 second in line and SHA-256 with the relatively highest time.
**Hashing function : SHA-256**
Filename: package_1:1KB
Hash Time : 14112 ns ,

**Hashing function : SHA-512**
Filename: package_1:1KB
Hash Time : 14403 ns,

**Hashing function : SHA-3-256**
Filename: package_1:1KB
Hash Time : 23122 ns ,

5.[2 points for 410/565] how performance of symmetric key encryption (AES), hash functions, and public-key encryption (RSA) compare to each other.
**Ans** :

- Symmetric Key encryption AES is very fast.
- Hashing produces a message digest and is comparatively having same timing results as symmetric key encryption.
- Public key encryption RSA on the other hand takes much more time than hashing and AES algorithm.

Below is a comparison from the experiment between symmetric key AES-CBC, hash and RSA for the same 1KB file called package_1
**AES-CBC 256-bit key 1KB file :**
KeyGen Time :4302 ns ,
Enc Time: 67625 ns,
Dec Time: 11306 ns ,
Encryption speed per byte: 67.625 ns,
Decryption speed per byte: 11.306 ns

**Hashing function : SHA-256**
Hash Time : 14112 ns ,
Per-Byte-Hash : 14.112 ns

**RSA 2048 bit key with 1KB file :**
KeyGen Time : 474146880 ns,
Enc Time: 5971043 ns,
Dec Time: 14925220 ns ,
Encryption speed per byte: 5971.043ns,
Decryption speed per byte: 14925.22ns

- AES key generation time is of the order of $10^6$ times less than key generation time of RSA key.
- Hashing produces a message digest and is comparatively having similar timing results as symmetric key encryption.
- AES encryption time is of the order of $10^2$ times less than encryption time using RSA. AES decryption is of the order of $10^3$ times less than decryption time using RSA.

**References:**


- COMPUTER SECURITY PRINCIPLES AND PRACTICE | William Stallings - Used to check about Public cryptography and encryption times in RSA.

- PyCryptodome official docs https://pycryptodome.readthedocs.io/en/latest/src/introduction.html - Used to compute various cryptography functions time
- PyDocs time library https://docs.python.org/3/library/time.html - Used for computation of nanoseconds on various functions.
- PyDocs base64 encoding https://docs.python.org/3/library/base64.html - Used for testing and checking encryption/decryption utf-8 ascii texts
- MB to bytes conversion - https://www.gbmb.org/bytes-to-mb - Used to get accurate bytes required.
- Dummy data generator - https://www.lipsum.com/ , https://superuser.com/questions/692175/how-to-create-a-random-txthuman-readable-text-like-ascii-file-in-linux, https://en.wikipedia.org/wiki/Public-key_cryptography, https://en.wikipedia.org/wiki/Digital_signature - Used to generate and fetch the data to fill for 1MB file , 10MB file for encryption/decryption