Homework 2

Akanksha Singh and Varun Rajeev Madathil

October 2, 2017

Abstract

This report consists of an analysis of different cryptographic schemes. The goal of this assignment in general was to use cryptographic libraries and then implement various cryptographic protocols. The protocols implemented, in particular, are AES-128 in the CTR mode, AES-256 in the CTR mode, RSA - 1024 encryption, RSA - 4096 encryption, HMAC MD5, HMAC SHA-1, HMAC-256 and digital signature with RSA-4096 and SHA-256. The overall goal of this assignment is to time the performance of these algorithms on a large file size (100 MB). This report shows the analysis of the algorithms. The algorithms are run a certain number of times, the average and median of the running times over these iterations are used to analyze the performance.

1 Introduction and preliminaries

In Java there are a lot of libraries that implement various cryptographic algorithms. The cryptographic algorithms that we are concerned about are the following AES-128 in the CTR mode, AES-256 in the CTR mode, RSA - 1024 encryption, RSA - 4096 encryption, HMAC MD5, HMAC SHA-1, HMAC-256 and digital signature with RSA-4096 and SHA-256.

AES stands for Advanced Encryption Standard and this is the standard that is used by most websites and security protocols for a secure encryption. This a symmetric key encryption and hence is expected to run very fast. We implement two versions of AES namely, AES-128 and AES-256. The main difference between the two is in the key size. The AES-128 implements 128 bit keys, where as the AES-256 implements 256-bit keys. AES-256 also has more rounds in comparison with AES-128. The CTR mode implies, that the block cipher mode of operation used is the counter mode.

RSA (Rivest - Shamir - Adleman) is an asymmetric key cryptosystem. This algorithm need not run very fast, since it involves very heavy underlying mathematical operations of exponentiation etc. There are two versions of RSA implemented as well. Namely, RSA-1024 and RSA-4096. The RSA-1024 is implemented using a 1024 bit key, and the RSA-4096 is implemented using a 4096 bit key. Here, we see that we cannot run the RSA algorithm on the entire data. We are required to run it in the ECB mode, which is basically the

Electronic Codebook mode. In this mode, the input message is divided into blocks and each of the blocks are encrypted separately. Now we see that the RSA encryption algorithm keeps a standard padding of 11 bytes. Therefore, the block size for our RSA algorithms will be (keysize - padding) in bytes. This is simply (1024/8 - 11) = 117 bytes for RSA-1024 and (4096/8 - 11) = 501 bytes for RSA - 4096. Hence, the messages were divided into chunks of 117 and 501 for RSA-1024 and RSA-4096 respectively and then encrypted.
HMAC is a type of message authentication code that uses a cryptographic hash function. The HMAC creates a hash digest for a message, that always has a fixed size. Three HMACs are implemented using different algorithms for the hash functions. MD5 stands for Message Digest 5, this creates a hash which has a length of 128 bits (16 bytes). HMAC using SHA-1 and SHA-256 are the next algorithms that are implemented. SHA stands for Secure Hash Algorithm. SHA-1 returns digests which are of size 20 bytes and SHA-256 returns digests of size 32 bytes.
Lastly, a digital signature scheme is implemented using the RSA digital signature scheme. Here the asymmetric key setting uses RSA-4096 and the hash algorithm used is SHA-256. Both of these algorithms have been introduced above. Here the message is hashed first and then encrypted using a private key. This is called signing. The signing times are calculated. When a user receives a signed message, he verifies it using the public key of the sender. The verification times of the program are measured as well.

2 Environment and testing details

The testing was done on the following environment and data. The screenshots of the results are shown in the next section.

- OS - macOS Sierra , Version 10.12.6
- Processor - 2.3 GHz Intel Core i5 (64 bit system)
- Memory - 8GB 2133 MHz LPDDR3
- Java version - Java 9
- File size - Text file of size 100 MB

Note : The submitted jar file has been compiled on the VCL image. This is just so that the evaluators do not face an issue while running the jar file on a Java 8 machine. The figures on this report are from my personal system that uses Java 9.

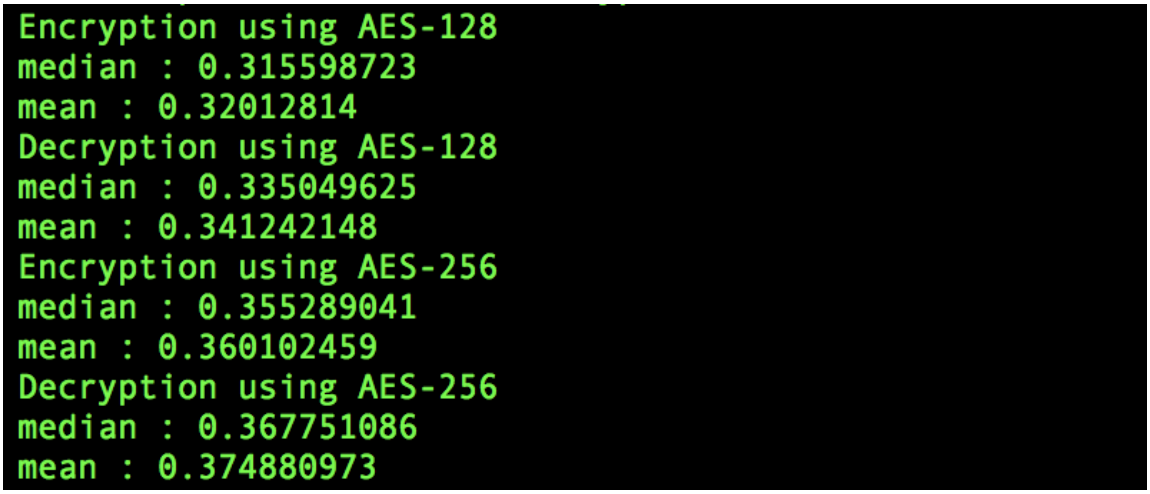

Figure 1: Mean and Median of AES

3 Results and Figures

Figure **??** shows the mean and median on AES 128 and AES 256 using the CTR mode. We see that on an average the AES-256 takes a little more time than AES-128, this can be attributed to the fact that the size of the keys and the number of rounds both are greater for AES-256 in comparison with AES-128.

Figure **??** shows the mean and median on RSA-1024 and RSA-4096.
The encryption and decryption times for RSA-4096 is greater than the encryption times and decryption times for RSA-1024. This can be attributed to the key size. We are doing operations over a larger modulus and computing higher powers in the case of RSA-4096 in comparison with RSA-1024. We also notice that the time required to do an encryption is far lesser than the time required to do a decryption. This is the case for both RSA-1024 and RSA-4096. This is because in RSA encryption, we choose a small number as our public exponent, which is a part our public key. Therefore, exponentiating a number by something like 3 takes far less time than exponentiating to something like 1024-bit or a 4096-bit number. Thus the encryption speeds are significantly faster than decryption speeds.

Figure **??** shows the mean and median on HMAC MD5, HMAC SHA-1 and HMAC SHA-256. HMAC using MD5 takes the least amount of time in comparison with the other Secure Hash Algorithms. We see that SHA-256 is slightly faster than SHA-1. This can be attributed to the fact that the number of rounds in SHA-1 is 80 and the number of rounds in SHA-256 is 64 where as the block sizes are the same on both the operations.

Figure **??** shows the mean and median on a digital signature scheme using RSA-4096 and SHA-256. The digital signature scheme takes time that is comparable to the symmet-

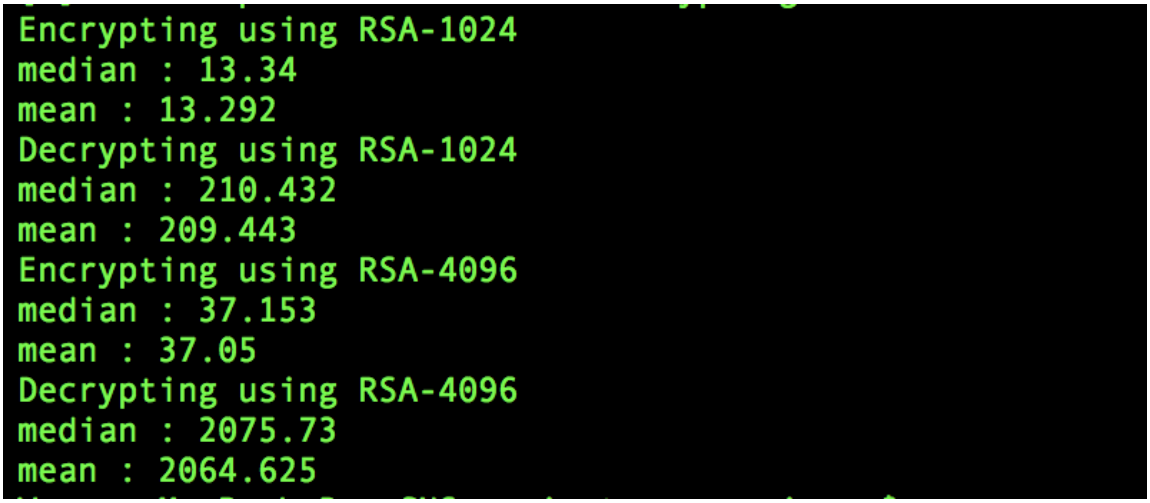

Figure 2: Mean and Median of RSA

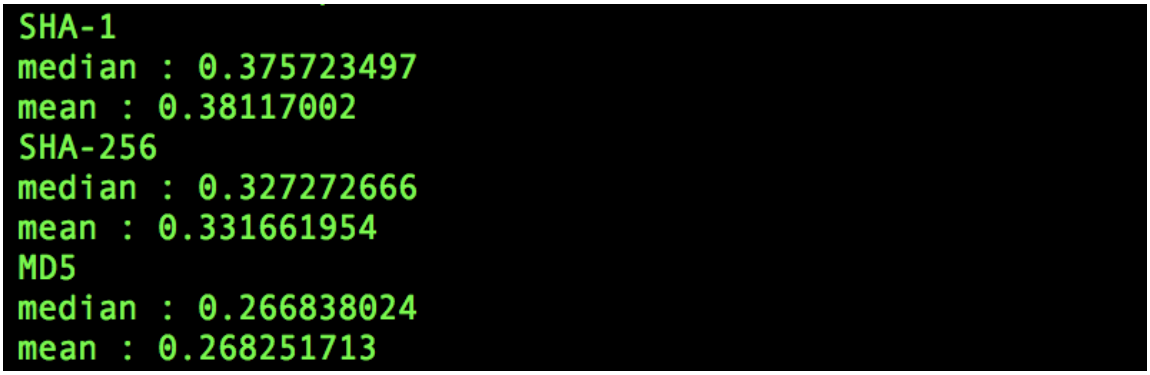

Figure 3: Mean and Median of HMAC

Figure 4: Mean and Median of Digital Signatures using RSA-4096 and SHA-256

ric key schemes. This is despite the fact the the signature scheme is asymmetric because, the encryption is done on the hash digests which are simply 256 bits long. Therefore even though we have an asymmetric key operation, the time it takes to create a signature is far less than the encryption schemes (which also involves asymmetric key operations).
Also, we notice that is SHA-256 is almost comparable with the digital signature implemented (we implement the digital signature in the code using the instance of SHA256withRSA, this implementation is optimized : there exists a DigestInfo structure in which the hash is encapsulated).

4 Algorithms and their median running times

Algorithm	Median Time (sec)
HMAC MD5	0.266
RSA Signature Signing	0.273
RSA Signature Verification	0.283
AES-128 Encryption	0.315
HMAC SHA-256	0.327
AES-128 Decryption	0.335
AES-256 Encryption	0.355
AES-256 Decryption	0.367
HMAC SHA-1	0.375
RSA-1024 Encryption	15.698
RSA-4096 Encryption	82.058
RSA-1024 Decryption	207.202
RSA-4096 Decryption	2525.536

5 Conclusions

Private key cryptography schemes use symmetric key (a single key) for encryption and decryption of a message whereas public key cryptography schemes use asymmetric keys (two keys). The public and private keys are very large to ensure that the computational difficulty on the scheme is hard to break. This can be achieved by large keys. This makes the computational time for public key schemes larger than private key schemes which can be seen in the table above. Public key schemes like RSA-1024 and RSA-4096 take large time compared to AES and HMAC. Digital signature takes less time even though they use RSA-4096 because we perform RSA on 256 bits (hash digest) in the signature scheme whereas for RSA-4096 encryption, the operations are done on a 100 MB file.

6 Acknowledgements

We would like to thank Prof. Brad Reaves for giving this opportunity to learn about the cryptographic protocols and their implementation. We thank him and the other Teaching Assistants, Sanghyun Ahn and Tae Hyun Kim for their guidance and supervision on the project. We also thank our friends who posted the many common queries on the Piazza board and their resolutions that helped us complete the project in a timely manner.