Samet Aytaç

2016700009

**CMPE526 TERM PROJECT REPORT**

In this project, I develop a program that encrypt and decrypt a file in a recursive manner. The goal of the project is simple: keeping user files secure. I use both symmetric and asymmetric encryption in this project.

I use symmetric encryption to encrypt-decrypt files. I use AES algorithm which is working ECB mode. It has fixed 128-bit key length. In encryption part, a random secret key is generated for file. All nested files and documents are encrypted with that secret key. Secret key which is encrypted with public key of user is also placed into file. Content of encrypted file can be seen in (1).

Esecret key(File), Epublic key(Secret Key) (1)

I use public-private key pairs for asymmetric encryption. These key pairs are used for keeping symmetric key secure. I use RSA algorithm for asymmetric encryption. It has 512-bit key length. User can have more than one key pairs. In this way, user can encrypt his/her files according to their confidentiality level. User can use different passwords for different keys. These key pairs are kept in user’s USB in encrypted way. Key pairs are encrypted with password. Content of USB can be seen in (2).

Epassword(Public Key), Epassword(Private Key) (2)

Notice that public key is also kept in encrypted way. It’s reason is ensuring message authentication. To encrypt file, user needs public key.(see (1)). To get public key, user needs to know password. In this way, only user who knows password can encrypt file.

Since symmetric key is encrypted with one of the user’s public key (user’s choice), user needs to have private key. To get private key from USB, user needs to know password. It means no one can decrypt an encrypted file other than user.

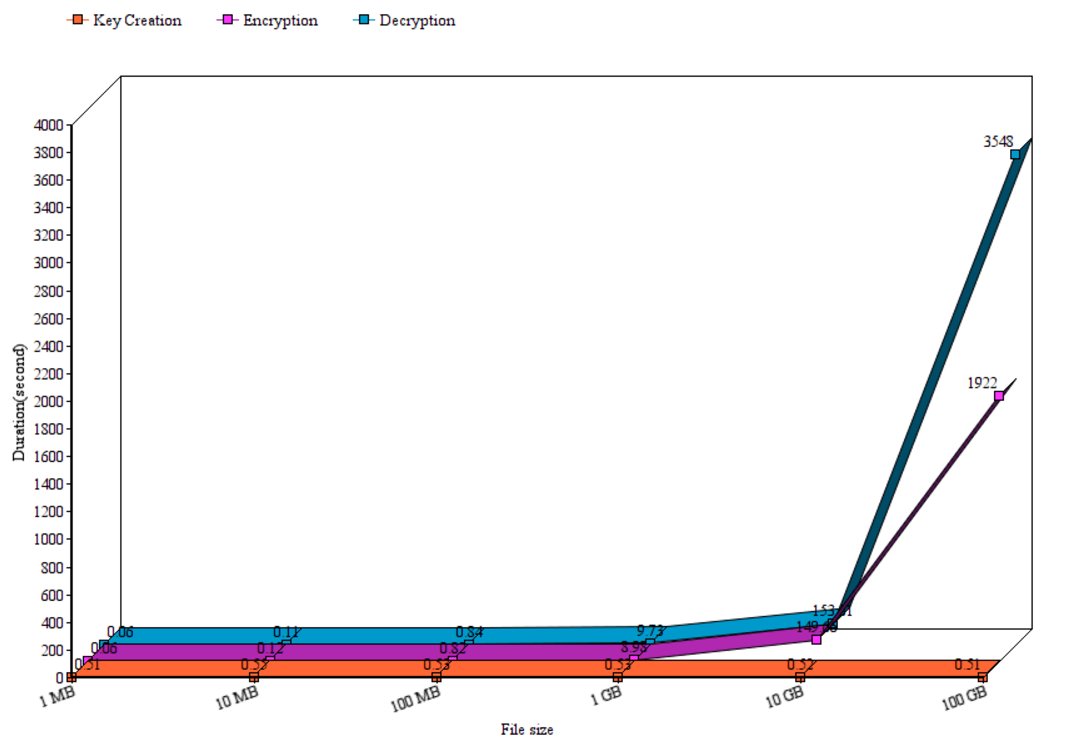
To sum up, a user needs to know password for both public key and private key. Thus, no one can encrypt or decrypt a file other than genuine user. This approach ensure both message authentication (no one can encrypt file) and confidentiality(no one can encrypt file).

I implement the program with Java. I use java.security library to implement asymmetric and symmetric encryption techniques. For asymmetric encryption functions, I modified an used some of the functions from [1].

Evaluation results can be found in Table-1 and Graph-1. I use Windows 10 operating system, Intel-i7-6700HQ 2.6 Ghz processor as test environment.

|  |  |  |  |
| --- | --- | --- | --- |
| File Size | Key Creation Duration | Encryption Duration | Decryption Duration |
| 1 MB | 0.51 s | 0.06 s | 0.06 s |
| 10 MB | 0.55 s | 0.12 s | 0.11 s |
| 100 MB | 0.53 s | 0.82 s | 0.84 s |
| 1GB | 0.53 s | 8.98 s | 9.73 s |
| 10 GB | 0.52 s | 149.66 s | 153.31 s |
| 100 GB | 0.51 s | 32.03 m | 59.13 m |

**Table 1: Performance according file size**

****

**Graph 1:** **Performance according file size**

As can be seen from graph-1 and table-1, key creation operation is not related with file size. Unlike key creation, encryption and decryption duration is related with file size. Encryption and decryption durations shows exponential-like behavior. Moreover, it can be seen that-especially for large file size- decryption of file is more costly than encryption.

**[1] https://www.mkyong.com/java/java-asymmetric-cryptography-example/**