uc3m | Universidad Carlos III de Madrid

BsC in Data Science and Engineering
2021-2022

Data protection & cybersecurity

"Lab of Ciphertext-Policy Attribute-Based Encryption"

Mohamed Afif Chifaoui

Rodrigo Oliver Coimbra

NIA (A): 100452024
NIA (R): 100451788

April 27, 2022

1 Introduction

For this practice we have implemented a Bash script named *Main.sh* that calls another Python script which first deploys the Attribute-Based Encryption implemented by CPABE and creates and the folders and structures necessary to carry out this activity. We also employed the statistical language R to obtain better insights into the relationships between the different variables in a clear and visual way.

The file *Encryption_Decription.py* is responsible for calling the methods defined in *ABE_Deployer.py* which prepare the environment and subsequently executes the functions that create the keys and encrypt and decrypt 20 times a given pdf file for a given user, in this case *user_0*. By doing this we will avoid the instabilities arising from the randomness of these measurements, working as some kind of mean.

The objetive of this practice is create a correct and relatively bug-free automatization of this process and to measure the time taken to carry out the different exercises so as to be able to reach some conclusions regarding the scalability of Attribute-Based Encryption depending on the number of users, attributes and file size.

NOTE: The scripts that automatized this process follow closely the indications given in both the tutorial session and the pdf fil which contains the statement of the practice.

2 Exercises

2.1 [Exercise 1] Deploy an ABE system (create the master secret key and issue keys for users) taking into consideration five attributes and five users. How large are the master and user's secret keys? How long does it take to encrypt and decrypt 20 times a pdf of 5MB?

In this case we have 5 users and 5 attributes. The information written in the *time.txt* file by the Bash script and the Python script tell us that:

```
3.12776780128479s
```

We can see that total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 3.128s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	638
master_key	83
user_0_priv	808

2.2 [Exercise 2] Repeat the initial experiment with 5 attributes and 20 users.

In this case we have 20 users and 5 attributes. The information written in the *time.txt* file by the Bash script and the Python script tell us that:

```
2.9842207431793213s
```

We can see that total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 2.984s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	630
master_key	86
user_0_priv	839

2.3 [Exercise 3] Repeat the initial experiment with 20 attributes and 5 users.

In this case we have 5 users and 20 attributes. The information written in the *time.txt* file by the Bash script and the Python script tell us that:

```
5.137879848480225s
```

We can see that total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 5.138s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	663
master_key	99
user_0_priv	3079

2.4 [Exercise 4] Repeat the initial experiment with 20 attributes and 20 users.

In this case we have 20 users and 20 attributes. The information written in the *time.txt* file by the Bash script and the Python script tell us that:

```
5.765616178512573s
```

We can see that total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 5.766s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	663
master_key	90
user_0_priv	3129

2.5 [Exercise 5] How does ABE scale with the number of attributes and users considered?

Attribute-Based Encryption (ABE onwards) is an interesting approach that encrypts based on the attributes that different users might have. The decryption of a ciphered text can only be performed by a user whose personal key matches the attributes of the ciphered text. This is really important as it is a safeguard against collision attacks, preventing any criptanalysis into the keys themselves.

After performing several experiments with ABE we have seen that its performance when encrypting and decrypting a 5 MB pdf file is not affected by the number of users which makes it readily usable for any cases where a large number of users is involved. As for the number of attributes, its performance did indeed worsen as the number of attributes grew. This is a drawback of ABE due to its complexity introduced by its Attribute-Based policy.

Here is a scatterplot between the number of attributes and the time elapsed in encrypting and decrypting 20 times a 5 MB pdf file:

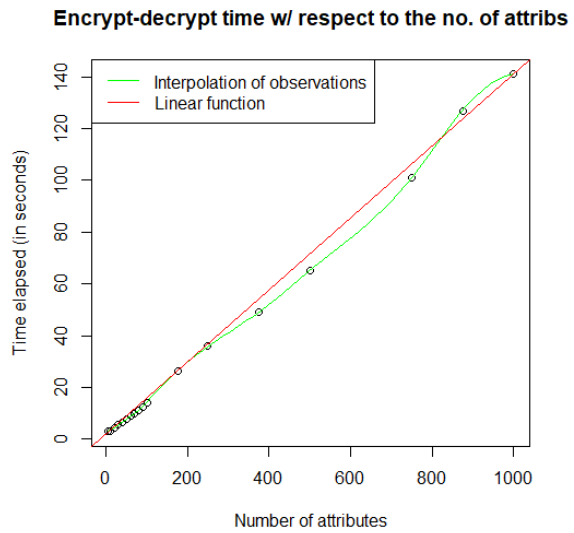

Figure 1: Encryption-decryption time as it varies with respect to the number of attributes.

It is clear the the time that it takes to encrypt and decrypt a file increases linearly with the number of attributes.

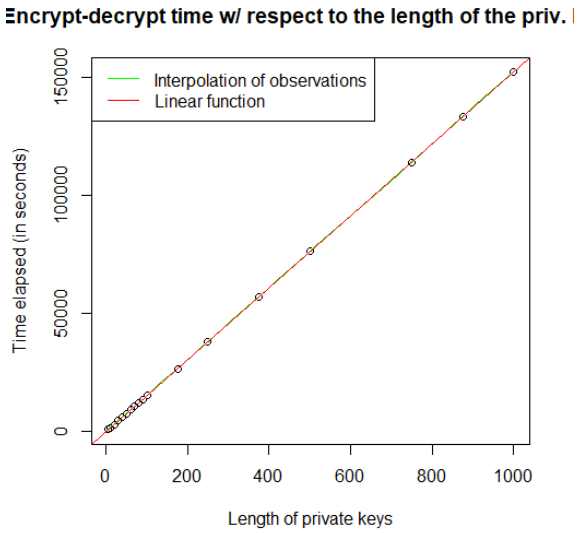

Figure 2: Encryption-decryption time as it varies with respect to the private user key length.

This case is even clearer: there almost a perfect match with a linear function when comparing the time that it takes to encrypt and decrypt a file with the length of the user (user_0's) private key. So much so, that the interpolation function and the linear function overlap in almost all the graph.

We would like to go further and analyze the correlation between the time elapsed, the number of attributes and the length of the private user key. *Note* that both the master key and public key lengths' remain stable throught all the modifications, independently of the number of attributes.

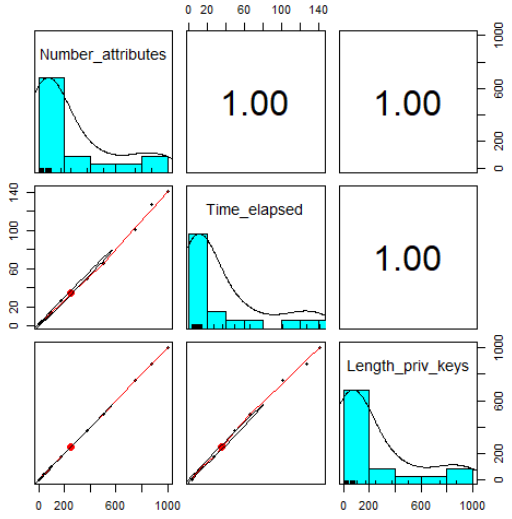

Figure 3: Correlation plot between the time elapsed, the number of attributes and the length of the private user key.

This shows that they are perfectly correlated and that their relationship is indeed entirely linear.

File size: We were also able to see that the encryption-decryption time of 5 MB, 10 MB and 20 MB pdf files *does* increase linearly with their size all else being equal (the number of users and

attributes).

Files used for this comparison:

* Document Centre Setup Guide for FlowPort (PDF, 5 MB)

* Cartographic Perspectives Number 13, Fall 1992 (PDF, 10 MB)

* Révision du genre Pilumnoides Lucas, 1844 (PDF, 20 MB)

2.6 [Exercise 6] What would you need to extrapolate your results to 1k users

If we have **1k users and 5 attributes** the *time.txt* file contains the following information:

```
2.8357810974121094s
```

which means that the total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 2.836s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	659
master_key	77
user_0_priv	859

If we have **1k users and 20 attributes** the *time.txt* file contains the following information:

```
4.755468368530273s
```

which means hat the total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 4.755s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	648
master_key	88
user_0_priv	3149

2.7 [Exercise 7] And what about 1k attributes? would it be feasible?

If we have **5 users and 1k attributes** the *time.txt* file contains the following information:

```
84.20065093040466s
```

which means hat the total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 84.201s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	649
master_key	93
user_0_priv	152488

If we have **20 users and 1k attributes** the *time.txt* file contains the following information:

```
82.8529965877533s
```

which means hat the total execution time for the encryption and decryption 20 times of a 5 MB document for *user_0* takes approximately 82.853s.

We opened the files *user_0_priv*, *pubKey*, *master_key* and *msk* with the text editor Vim and used the command

```
: !wc -m %
```

to obtain the number of characters per file.

Length of key files	
File name	Length (in characters)
pubKey	631
master_key	89
user_0_priv	152368

Assessment: it is clear that this is *not* feasible for a production environment; for example, with 5 users and 1k attributes 20 encryption-decryption cycles took around 84s to be finished which means that $\frac{84s}{20loops} = 4.2\text{s}\ loop^{-}1$. This is clearly too slow for any real-world application where speed is minimally important, as both the security and usefulness of the data often lie on in the immediacy of its reception.

3 Conclusions

From this task regarding attribute based encryption we drew several conclusions. One of these conclusions is to see how the time evolves when increasing either the number of attributes, the number of users or the file size depending on the experiment we were deploying. Also, we have learnt a lot of new things while implementing this code and executing it into the bash terminal that seemed a bit challenging at first glimpse, but fortunately we reached to get the required questions that we have been discussing through this report.

We got valuable insight from the implementation of Attribute-Based Encryption and gained useful practical knowledge about its behavior and properties. It was also a bonus to be able to learn more about the Debian GNU/Linux operating system.

We are looking forward to learn more about cybersecurity and data protection.