Smit Patel

M08582121

Charles Greenman

M08752091

Project 3

Operating System:

Windows 10 (Intel Core I7) & Arch Linux 5.2.10 (Intel i5)

Language:

We used python 3.7 IDE to compile and run our code in the command prompt with the example instructions given from the project3.pdf

Libraries:

We used pycryptodome, hashlib and imported Json

Compile:

In order to compile this code, we need to make sure, the root folder in the command prompt and use the main.py to run it with right file names for each of the 4 functions used. (use the highlighted file names for the function)

Key Generation Function:

We are using SHA256 to simulate the PRF, therofore, no skprf.txt file is required as there is no key for that algorithm.

Root the folder in the command prompt and compile by **"python main.py keygen ../data/skaes.txt"**

Output:

```
conda : The term 'conda' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spe
included, verify that the path is correct and try again.
At line:1 char:1
+ conda activate base
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121> python main.py  keygen ../data/skaes.txt
C:\Python36\python.exe: can't open file 'main.py': [Errno 2] No such file or directory
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121> cd src
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src> python main.py  keygen ../data/skaes.txt
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src>
```

skaes.txt - Notepad

File Edit Format View Help

```
94fc8b60c36272bd6c2b4dc0fb71349950f4047c5ae9cb210c02d3276777fe56
```

Encryption Function:

Root the folder in the command prompt and compile by **"python main.py enc ../data/skaes.txt ../data/index.txt ../data/files ../data/ciphertextfiles"**

Output:

```
C:\Python36\python.exe: can't open file 'main.py': [Errno 2] No such file or directory
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121> cd src
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src> python main.py  keygen ../data/skaes.txt
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src> python main.py enc ../data/skaes.txt ../data/index.txt ../data/files ../data/ciphertext
files
Processed file: f1.txt
Processed file: f2.txt
Processed file: f3.txt
Processed file: f4.txt
Processed file: f5.txt
Processed file: f6.txt
Encrypted Index
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src>
```

Indexfile Generated:

index.txt - Notepad

File Edit Format View Help

c2.txt", "c3.txt", "c5.txt"], "462aec2ca8883242483416b27326b2258b9ae4d116aef0c55c38b1820d07355e": ["c2.txt"], "e3b0c44298fc1c149afbf4c8996fl

Ciptertextfiles Generated in the folder:

c1.txt	12/2/2019 10:44 PM	Text Document	1 KB
c2.txt	12/2/2019 10:44 PM	Text Document	1 KB
c3.txt	12/2/2019 10:44 PM	Text Document	1 KB
c4.txt	12/2/2019 10:44 PM	Text Document	1 KB
c5.txt	12/2/2019 10:44 PM	Text Document	1 KB
c6.txt	12/2/2019 10:44 PM	Text Document	1 KB

Token Function:

Root the folder in the command prompt and compile by **"python main.py token packers ../data/token.txt"**

Output:

```
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src> python main.py token packers  ../data/token.txt
Input Token: packers
Output Hash: 81ac4a9305a1e8b96418b7b444aa171586bbf8697b84768a0f11dddfccd48533
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src>
```

token.txt - Notepad

File Edit Format View Help

```
81ac4a9305a1e8b96418b7b444aa171586bbf8697b84768a0f11dddfccd48533
```

Search Function:

Root the folder in the command prompt and compile by **"python main.py search ../data/index.txt ../data/token.txt ../data/ciphertextfiles ../data/skaes.txt"**

Output:

```
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src> python main.py search ../data/index.txt ../data/token.txt ../data/ciphertextfiles ../data/skaes.txt
Files matching token 81ac4a9305a1e8b96418b7b444aa171586bbf8697b84768a0f11dddfccd48533: c2.txt, c3.txt, c5.txt
$c2.txt decrypted contents are: $b'packers patriots '
$c3.txt decrypted contents are: $b'packers'
$c5.txt decrypted contents are: $b'steelers packers'
PS C:\Users\Jsmit_1n8uqvk\Desktop\Data Security and Privacy\se_m08582121\src>
```

***The associated Token Files are decrypted (above).**

Benchmarking The Program

We measured the performance of building the encrypted index, creating the token and searching for the token using the Unix time command, here are the results, this was performed on a machine running Arch Linux.

```
[chuck@chuck src]$ time python main.py enc ../data/skaes.txt ../data/index.txt ../data/files/ .
./data/ciphertextfiles/
Processed file: f6.txt
Processed file: f5.txt
Processed file: f3.txt
Processed file: f1.txt
Processed file: f2.txt
Processed file: f4.txt
Encrypted Index

real    0m0.137s
user    0m0.130s
sys     0m0.007s
```

```
[chuck@chuck src]$ time python main.py token packers ../data/token.txt
Input Token: packers
Output Hash: 81ac4a9305a1e8b96418b7b444aa171586bbf8697b84768a0f11dddfccd48533

real    0m0.128s
user    0m0.117s
sys     0m0.010s
[chuck@chuck src]$
```

```
[chuck@chuck src]$ time python main.py search ../data/index.txt ../data/token.txt ../data/ciphe
rtextfiles/ ../data/skaes.txt
Files matching token 81ac4a9305a1e8b96418b7b444aa171586bbf8697b84768a0f11dddfccd48533: c5.txt,
c3.txt, c2.txt
$c5.txt decrypted contents are: $b'steelers packers'
$c3.txt decrypted contents are: $b'packers'
$c2.txt decrypted contents are: $b'packers patriots '

real    0m0.136s
user    0m0.116s
sys     0m0.020s
```

Explaining The Implementation of this Searchable Encryption Scheme

Here is a high level overview of how this system works:

1. We generate a key for the AES function.
2. Before encrypting the files, we created hashes of the files using SHA256, which serves as the index for our search.
3. We encrypt all files with the AES function and key.
4. When accepting a search query we use SHA256 to hash the query and search the hashed index, when matches are found we use the AES key to decrypt the full file.