

Report on Searchable Encryption

Mitra Sasanka Darbha, M#13523468

Anish Ghantasala, M#13518394

1. Introduction

SEARCHABLE ENCRYPTION:

In searchable encryption, client encrypts all the files, keeps secret key and uploads encrypted files to server.

Pros:

- Data and queries are private
- Server can search without knowing anything
- Client retrieves only matched files for a keyword
- Client saves storage and querying

The disadvantage of using Searchable Encryption is that the search time is slower but it's still practical.

INVERTED INDEX:

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. In simple words, it is a HashMap like data structure that directs you from a word to a document or a web page. The advantage of Inverted Index is that it allows fast full text searches and is easy to develop. However, large storage head and high maintenance cost on update, delete and insert.

AES ENCRYPTION:

AES is based on 'substitution-permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). The number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

2. Environment

The following configuration is used in this project:

Processor	AMD A8 64-bit 2.20GHz
RAM	8 GB (6.94 usable)
Operating System	Microsoft© Windows® 10 Home Single Language
Language	Python 3.7.4 (64-bit) with IDLE.
Libraries used	os, sys, random, timeit, Crypto.Cipher

3. Execution and Results

Name of the project folder is **se_m13523468**. It contains two subfolders – **src**, **data** – and the **report.pdf** file. The source files reside under **src** folder and the files required and/or generated by the programs reside under **data** folder. This folder also contains the screenshots of the program execution.

Since Python Language is used for programming, there is no **build** folder as there will be no .cpp or .h or .class files. Command Prompt is used for the execution of python files.

Details of the individual files in the folder:

File Name	Description	Path
<i>keygen_se.py</i>	To generate PRF and AES keys	se_m13523468\src\
<i>inv_ind_enc.py</i>	To generate encrypted inverted index and encrypt the files	
<i>token_gen_se.py</i>	To generate a token	
<i>inv_ind_search.py</i>	To search for token in encrypted inverted index	
<i>index.txt</i>	Contains the encrypted inverted index	
<i>iv.txt</i>	Contains 128-bit IV for AES encryption	se_m13523468\data\
<i>result.txt</i>	Contains result of search	
<i>skaes.txt</i>	Contains 256-bit AES key	
<i>skprf.txt</i>	Contains 256-bit PRF key	
<i>token.txt</i>	Contains encrypted search keyword	
<i>files</i>	Contains all files in plaintext	
<i>ciphertextfiles</i>	Contains all corresponding encrypted files	

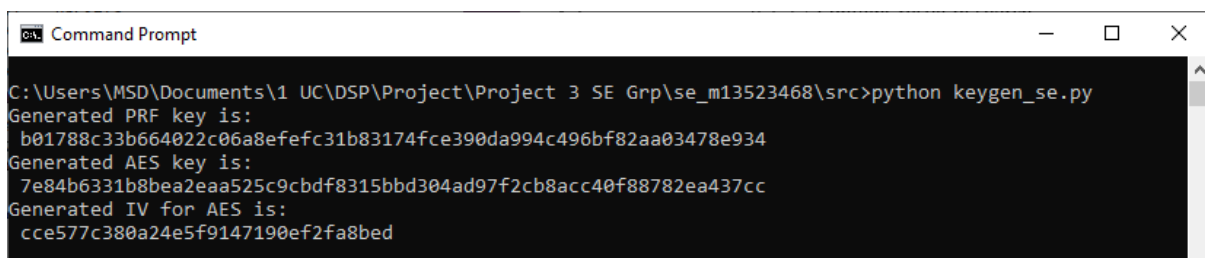
Key Generation Function:

keygen_se.py is executed to generate random 256-bit keys for PRF and AES. Also, a 128-bit IV is also generated for AES_CBC_256. These keys are stored in *skprf.txt*, *skaes.txt*, *iv.txt* respectively.

The command to generate keys is:

C: ~\se_m13523468\src > python keygen_se.py

Execution is as shown:



```
Command Prompt
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 3 SE Grp\se_m13523468\src>python keygen_se.py
Generated PRF key is:
b01788c33b664022c06a8efefc31b83174fce390da994c496bf82aa03478e934
Generated AES key is:
7e84b6331b8bea2eaa525c9cbdf8315bbd304ad97f2cb8acc40f88782ea437cc
Generated IV for AES is:
cce577c380a24e5f9147190ef2fa8bed
```

Encryption Function:

inv_ind_enc.py reads the secret keys and IV for PRF and AES encryption from skprf.txt, skaes.txt, iv.txt respectively. It also accepts two folder paths – one with files in plain text, other for storing encrypted files.

timeit module is used to calculate the time of execution of this program which includes – generation of encrypted inverted index and also encrypting all the files in the given folder.

Files are labelled as f1.txt, f2.txt ... f6.txt, which are encrypted as c1.txt, c2.txt ... c6.txt respectively using AES-CBC-256

Files contain the following keywords:

f1.txt: bengals steelers packers
f2.txt: packers patriots
f3.txt: packers
f4.txt: steelers bengals
f5.txt: steelers packers
f6.txt: bengals

Encrypted Inverted Index Generation:

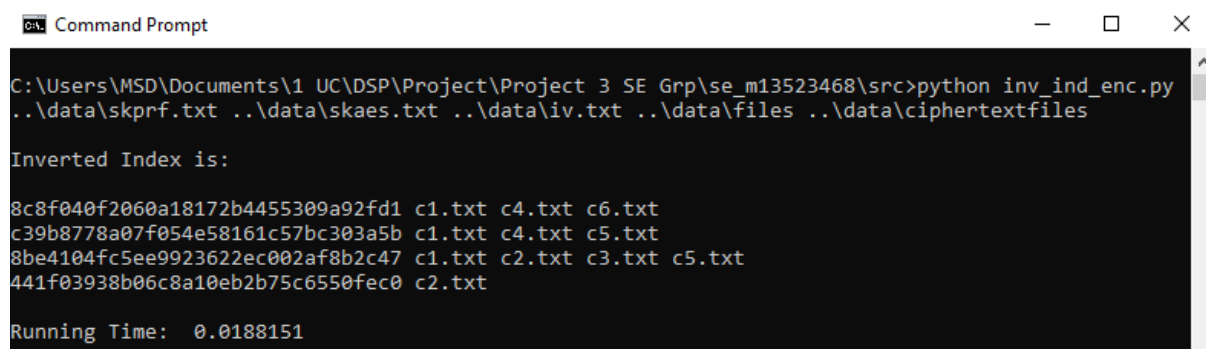
A plain inverted index is generated first. Python Dictionary and List data structures are used to create Inverted Index. Keywords are taken from each and every file and these keywords are considered to be “keys” of the dictionary. Each “value” of a key is a list containing files with corresponding keyword.

Once such a plain inverted index is generated, all the keywords are encrypted with PRF (AES-ECB-256) and files are encrypted with AES-CBC-256. A similar data structure is used to store this data. Such a data structure is called the Encrypted Inverted Index. This is stored in index.txt.

The command to generate encrypted inverted index is:

```
C: ~\se_m13523468\src > python inv_ind_enc.py ..\data\skprf.txt ..\data\skaes.txt ..\data\iv.txt  
..\data\files ..\data\ciphertextfiles
```

Execution is as shown:



```
Command Prompt  
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 3 SE Grp\se_m13523468\src>python inv_ind_enc.py  
..\data\skprf.txt ..\data\skaes.txt ..\data\iv.txt ..\data\files ..\data\ciphertextfiles  
  
Inverted Index is:  
  
8c8f040f2060a18172b4455309a92fd1 c1.txt c4.txt c6.txt  
c39b8778a07f054e58161c57bc303a5b c1.txt c4.txt c5.txt  
8be4104fc5ee9923622ec002af8b2c47 c1.txt c2.txt c3.txt c5.txt  
441f03938b06c8a10eb2b75c6550fec0 c2.txt  
  
Running Time: 0.0188151
```

Running Time obtained is **0.018 secs**

Token Generation Function:

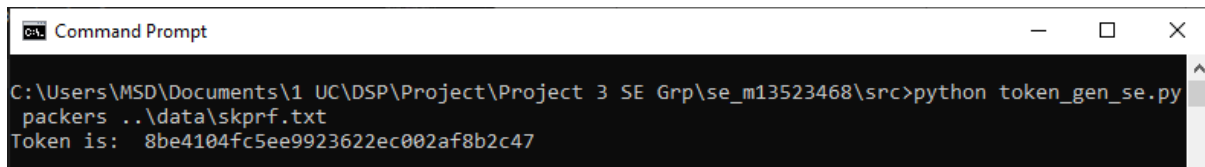
token_gen_se.py reads keyword and also the secret key for PRF encryption from skprf.txt. It encrypts the keyword with PRF and writes the token to token.txt in hexadecimal format.

The command to generate token is:

```
C: ~\se_m13523468\src > python token_gen_se.py packers ..\data\skprf.txt
```

Here token is generated for the keyword – packers.

Execution is as shown:



```
Command Prompt
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 3 SE Grp\se_m13523468\src>python token_gen_se.py
packers ..\data\skprf.txt
Token is: 8be4104fc5ee9923622ec002af8b2c47
```

Search Function:

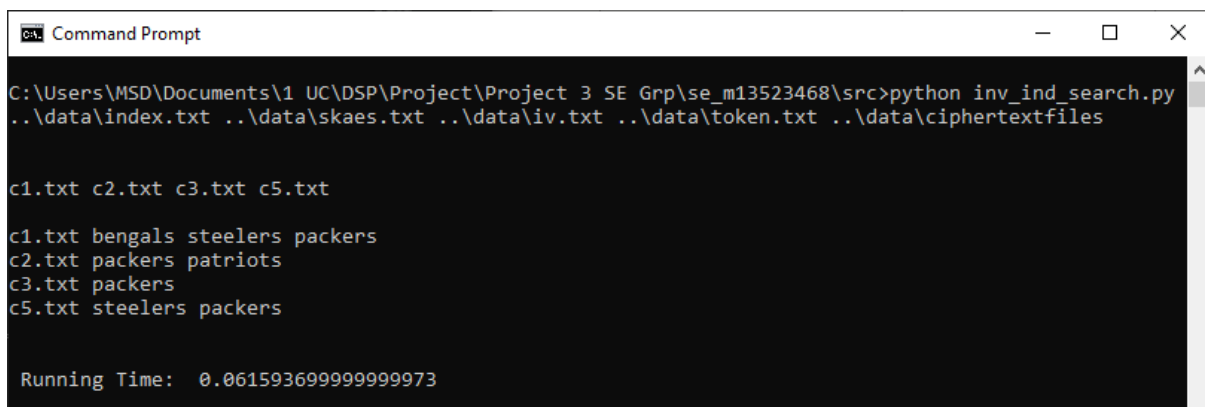
inv_ind_search.py reads AES secret key and IV from skaes.txt and iv.txt respectively. It also reads the token from token.txt and encrypted inverted index from index.txt. The path of cipher text files is also given as input.

When the token is found in the encrypted inverted index, the corresponding files are decrypted, and the data is displayed. If token not found, then corresponding message is displayed. The result of this search is stored in result.txt.

The command to search a token is

```
C: ~\se_m13523468\src > python inv_ind_search.py ..\data\index.txt ..\data\skaes.txt
..\data\iv.txt ..\data\token.txt ..\data\ciphertextfiles
```

The execution is as shown:



```
Command Prompt
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 3 SE Grp\se_m13523468\src>python inv_ind_search.py
..\data\index.txt ..\data\skaes.txt ..\data\iv.txt ..\data\token.txt ..\data\ciphertextfiles

c1.txt c2.txt c3.txt c5.txt
c1.txt bengals steelers packers
c2.txt packers patriots
c3.txt packers
c5.txt steelers packers

Running Time: 0.061593699999999973
```

Running time obtained is **0.061 secs**

4. References

1. https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm
2. <https://www.geeksforgeeks.org/inverted-index/>
3. <https://docs.python.org/3/tutorial/datastructures.html>