# SE Project Report

## How to Run:

1. May need to install the pycryptodome library
    a. pip install pycryptodome
2. Run the file "se_m12804663/src/ SearchableEncryption.py"
    a. This will run through the KeyGen, Enc, TokenGen, and Search functions with default values and should output a combination of the pictures below
3. If you would like to pass inputs, there are four options:
    a. KeyGen
        i. SearchableEncryption.py KeyGen
        ii. SearchableEncryption.py KeyGen 256 ../data/skprf.txt ../data/skaes.txt
    b. Enc
        i. SearchableEncryption.py Enc
        ii. SearchableEncryption.py Enc ../data/skprf.txt ../data/skaes.txt ../data/index.txt ../data/files ../data/ciphertextfiles
    c. TokenGen
        i. SearchableEncryption.py TokenGen
        ii. SearchableEncryption.py TokenGen packers ../data/skprf.txt ../data/token.txt
    d. Search
        i. SearchableEncryption.py Search
        ii. SearchableEncryption.py Search ../data/index.txt ../data/token.txt ../data/ciphertextfiles ../data/skaes.txt ../data/results.txt

## Description:

For this project, I decided to write the searchable encryption code in the latest version of Python within Visual Studio 2022 and worked in Windows OS. I used the latest pycryptodome library to implement the AES encryption and decryption.

### KeyGen:

This function generates two secret keys given a length parameter, one for PRF and one for AES, and writes it to "../data/skprf.txt" and "../data/skaes.txt".

I programmed this by first dividing the given length by 8 to get bytes instead of bits for encryption purposes. Then I generated random sets of bytes for the prf and aes keys using the adjusted length and wrote each of these keys to the associated files

```
prfKey: b'\x12\x8e[D\x99\x17`\x84!\xd6|\xa4\x16\x0eA-\xfb\xaao,\xa6\x99L8\xd2\xc7\xd3\xb8\xac\x92\xb3\xae'
aesKey: b'2\xf2\xaeU\xd4\x97\xbf\xf2b\x0c@%1\xe00\xe65\x8e\x82B\xa2j\x8bm\x07_\xe3%M-\xab\t'
```

## Enc:

This function reads the data from the files and the secret keys from the key files to build an inverted index or a keyword-file matrix. It then writes the index to the file "../data/index.txt".

I programmed this by retrieving the keys from the prf and aes key files. Then I went through each file in the files directory and extracted each keyword from each file. These keywords were then encrypted with the encryption cipher of the prf key. Then they were added to a list of the cipher keywords. Next, all the files were encrypted and written to their associated cipher file in the given directory. After this, I created a 2D index and added the cipher keywords to each row. Then for each file, I marked in the index if a key is in the file by listing the related ciphertext file. Finally, I wrote the generated index to the index file and also printed it to the console.

```
Index
[b'sqezhCEOCuyQSFXu51ZO5g==', 'c1.txt', 0, 0, 'c4.txt', 0, 'c6.txt']
[b'/otXJmH92OcclmmX40/uKw==', 'c1.txt', 0, 0, 'c4.txt', 'c5.txt', 0]
[b'6UhkNPOe9ijCphTmAIbTJQ==', 'c1.txt', 'c2.txt', 'c3.txt', 0, 'c5.txt', 0]
[b'/CJw1sQWrkm4K+VS149UUA==', 0, 'c2.txt', 0, 0, 0, 0]
```

## TokenGen:

This function reads a secret prf key and generates a token using the key. Then it writes the token to the file "../data/token.txt".

I developed this by opening the prf file to retrieve the prf key. I used the prf key to encrypt the given keyword into a token. Then I printed the token to the console and wrote it to the token file.

```
GENERATED TOKEN: b'6UhkNPOe9ijCphTmAIbTJQ=='
```

## Search:

This function reads a token from the token file and the index from the index file to find the associated files with the token over the index. It then prints the file identifiers and the decrypted contents of the files along with writing this to the file "../data/results.txt".

I created this by retrieving the token from the token file and the index from the index file. Then I searched the index and obtained the row from the matched token and got each associated cipher file. Next, I decoded the associated cipher files and printed each file's decrypted contents. Then I wrote it to the results file. If no token was found, or if the token was not in the cipher files, then the results file will be empty, and the console will print a similar statement.

```
c1.txt c2.txt c3.txt c5.txt

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