Report on
Symmetric-Key Encryption

By Mitra Sasanka Darbha, M#13523468

1. Introduction

ADVANCED ENCRYPTION STANDARD:

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).

AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix . 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.

CIPHER BLOCK CHAINING:

In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.

2. Environment

The following system configuration is used in this project.

Processor	AMD A8 64-bit 2.20GHz
Operating System	Microsoft© Windows® 10 Home Single Language
Language	Python 3.7.4 (64-bit) with IDLE.

A third-party library "pycrypto" is used to implement AES. It can be obtained by using the following command in Windows Command Prompt:

```
C:\> python -m pip install --user pycrypto
```

Visual C++ 14.0 must also be installed as it is a dependency for pycrypto package.

If Linux system is used, the command is **pip install pycrypto** in terminal.

3. Execution and Results

Name of the project folder is **aes_m13523468.** It contains two subfolders – **src** and **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.

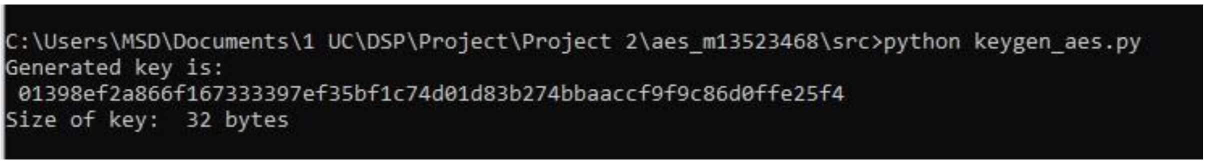
Details of the individual files in the folder:

File Name	*Description*	*Path*
encrypt_aes.py	To encrypt plaintext using AES.	aes_m13523468\src\
decrypt_aes.py	To decrypt ciphertext using AES.	
keygen_aes.py	To generate random secret key of 256 bits.	
enc55.py	To encrypt text in both CBC and ECB Modes.	
enc_dec_time.py	To calculate time for encryption & decryption.	
plaintext.txt	Contains the text that is encrypted by AES.	aes_m13523468\data\
key.txt	Contains 256-bit Secret Key for AES.	
iv.txt	Contains 128-bit Initialization Vector for AES.	
ciphertext.txt	Contains cipher obtained after encryption.	
result.txt	Contains plaintext decrypted by AES.	

Key Generation Function:

AES-CBC-256 uses a 256-bit key. So, a random key of 32 bytes (32*8=256bits) is generated.

keygen_aes.py is executed on command line as shown. The output is a 256bit random key in hexadecimal format. This result is also stored in key.txt in **data** folder.



```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>python keygen_aes.py
Generated key is:
 01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
Size of key:  32 bytes
```

Encryption:

AES-CBC-256 encryption uses a 16-byte Initialization Vector. This IV is also randomly generated and stored in iv.txt as hexadecimal format in **data** folder.

encrypt_aes.py reads the plain text that needs to be encrypted from plaintext.txt. It reads the secret key from key.txt and encrypts the plain text using key and iv. Padding is used such that block size is a multiple of 16.

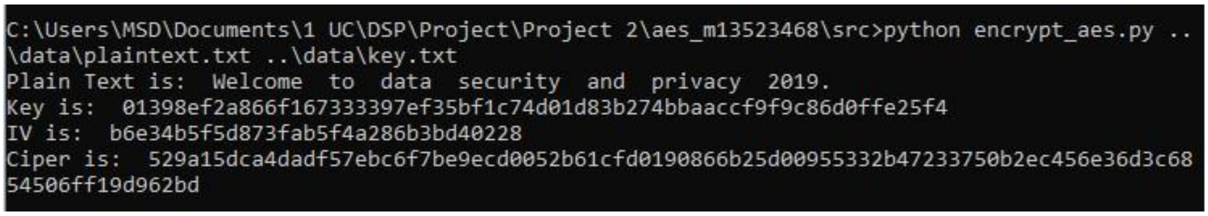
The cipher text is stored in ciphertext.txt as hexadecimal format in **data** folder.

In Command Prompt, traverse to the **src** folder in **aes_m13523468.** In my system it is C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>

The command to perform encryption is:

C: ~\aes_m13523468\src > python encrypt.py ..\data\plaintext.txt ..\data\key.txt

The execution is shown:



```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>python encrypt_aes.py ..
\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  b6e34b5f5d873fab5f4a286b3bd40228
Ciper is:  529a15dca4dadf57ebc6f7be9ecd0052b61cfd0190866b25d00955332b47233750b2ec456e36d3c68
54506ff19d962bd
```

Decryption:

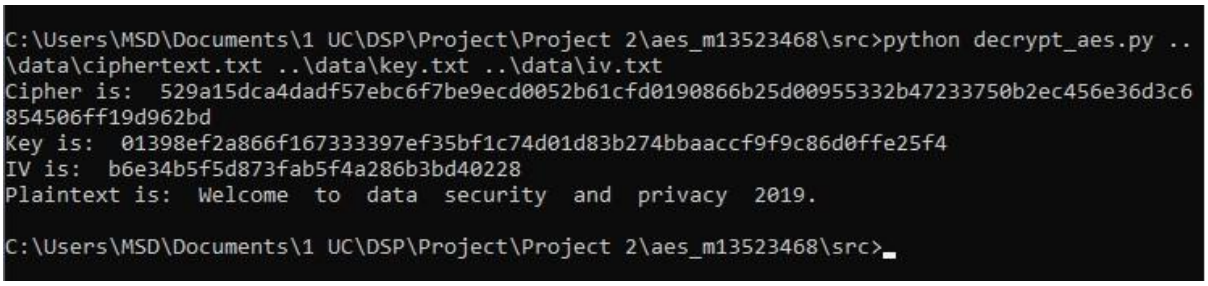
decrypt_aes.py reads Initialization Vector from iv.txt, key from key.txt, and cipher from ciphertext.txt. The result is stored in result.txt in **data** folder as human-readable format.

In Command Prompt, traverse to the **src** folder in **aes_m13523468.** In my system it is C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>

The command to perform decryption is:

C: ~\aes_m13523468\src > python decrypt.py ..\data\ciphertext.txt ..\data\key.txt ..\data\iv.txt

The execution is as shown:

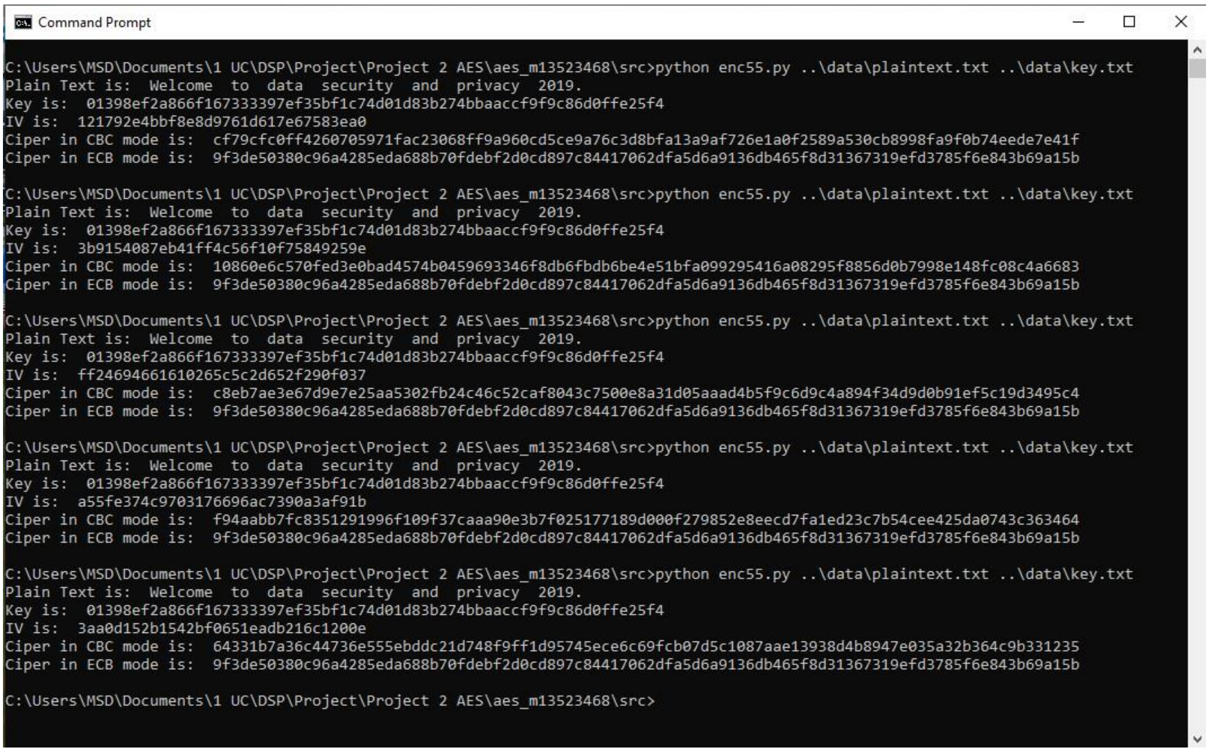


```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>python decrypt_aes.py ..
\data\ciphertext.txt ..\data\key.txt ..\data\iv.txt
Cipher is:  529a15dca4dadf57ebc6f7be9ecd0052b61cfd0190866b25d00955332b47233750b2ec456e36d3c6
854506ff19d962bd
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  b6e34b5f5d873fab5f4a286b3bd40228
Plaintext is:  Welcome  to  data  security  and  privacy  2019.

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2\aes_m13523468\src>
```

Different Modes of Encryption:

The plaintext is encrypted in two modes – ECB and CBC, and the program enc55.py is run for 5 times. The command to execute this code is:

C: ~\aes_m13523468\src > python enc.py ..\data\plaintext.txt ..\data\key.txt



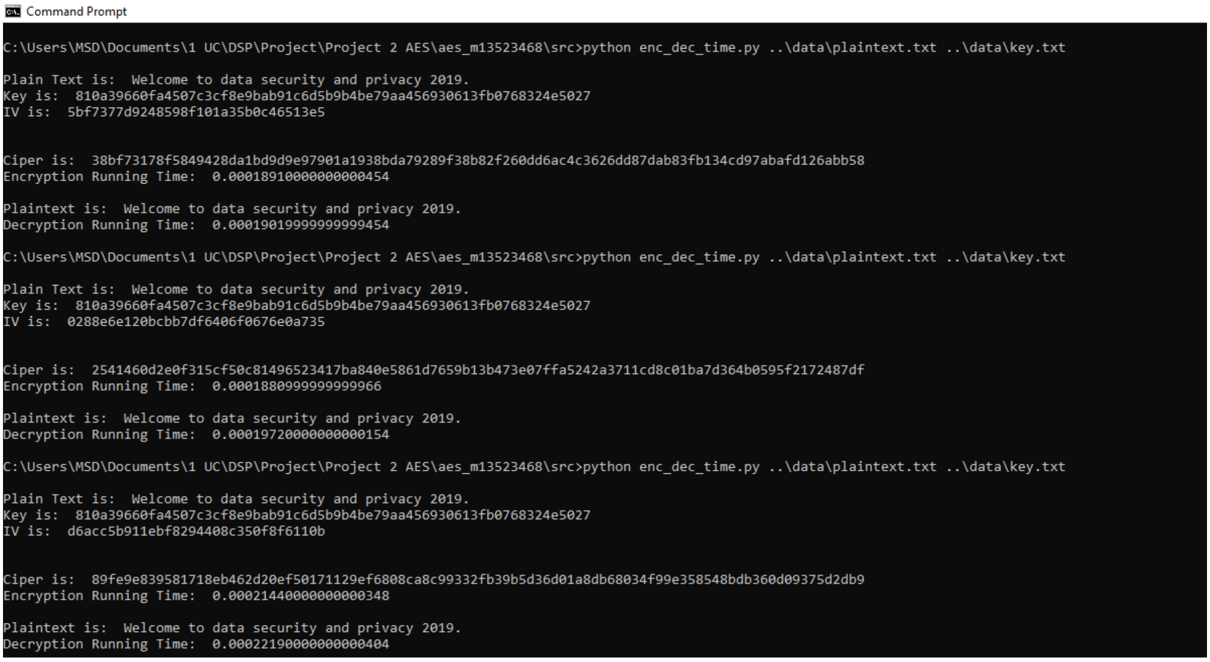
```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc55.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  121792e4bbf8e8d9761d617e67583ea0
Ciper in CBC mode is:  cf79cfc0ff4260705971fac23068ff9a960cd5ce9a76c3d8bfa13a9af726e1a0f2589a530cb8998fa9f0b74eede7e41f
Ciper in ECB mode is:  9f3de50380c96a4285eda688b70fdebf2d0cd897c84417062dfa5d6a9136db465f8d31367319efd3785f6e843b69a15b

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc55.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  3b9154087eb41ff4c56f10f75849259e
Ciper in CBC mode is:  10860e6c570fed3e0bad4574b0459693346f8db6fbdb6be4e51bfa099295416a08295f8856d0b7998e148fc08c4a6683
Ciper in ECB mode is:  9f3de50380c96a4285eda688b70fdebf2d0cd897c84417062dfa5d6a9136db465f8d31367319efd3785f6e843b69a15b

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc55.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  ff24694661610265c5c2d652f290f037
Ciper in CBC mode is:  c8eb7ae3e67d9e7e25aa5302fb24c46c52caf8043c7500e8a31d05aaad4b5f9c6d9c4a894f34d9d0b91ef5c19d3495c4
Ciper in ECB mode is:  9f3de50380c96a4285eda688b70fdebf2d0cd897c84417062dfa5d6a9136db465f8d31367319efd3785f6e843b69a15b

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc55.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  a55fe374c9703176696ac7390a3af91b
Ciper in CBC mode is:  f94aabb7fc8351291996f109f37caaa90e3b7f025177189d000f279852e8eecd7fa1ed23c7b54cee425da0743c363464
Ciper in ECB mode is:  9f3de50380c96a4285eda688b70fdebf2d0cd897c84417062dfa5d6a9136db465f8d31367319efd3785f6e843b69a15b

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc55.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  Welcome  to  data  security  and  privacy  2019.
Key is:  01398ef2a866f167333397ef35bf1c74d01d83b274bbaaccf9f9c86d0ffe25f4
IV is:  3aa0d152b1542bf0651eadb216c1200e
Ciper in CBC mode is:  64331b7a36c44736e555ebddc21d748f9ff1d95745ece6c69fcb07d5c1087aae13938d4b8947e035a32b364c9b331235
Ciper in ECB mode is:  9f3de50380c96a4285eda688b70fdebf2d0cd897c84417062dfa5d6a9136db465f8d31367319efd3785f6e843b69a15b

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>
```

It can be observed that the cipher text generated remains the same for all executions in ECB Mode. However, cipher texts differ in CBC Mode. This is because there is no usage of IV in ECB mode. That is, the cipher text is deterministic for a pair of key and plaintext.

In CBC Mode, a random Initialization Vector IV is used. Since IV is randomly generated, CBC mode results in different ciphers for same set of key and plaintext. By this we can deduce the fact that CBC Mode provides more security than ECB Mode.

Average Running Time:

The average running time for AES CBC encryption and decryption is as shown:



```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc_dec_time.py ..\data\plaintext.txt ..\data\key.txt

Plain Text is:  Welcome to data security and privacy 2019.
Key is:  810a39660fa4507c3cf8e9bab91c6d5b9b4be79aa456930613fb0768324e5027
IV is:  5bf7377d9248598f101a35b0c46513e5


Ciper is:  38bf73178f5849428da1bd9d9e97901a1938bda79289f38b82f260dd6ac4c3626dd87dab83fb134cd97abafd126abb58
Encryption Running Time:  0.00018910000000000454

Plaintext is:  Welcome to data security and privacy 2019.
Decryption Running Time:  0.00019019999999999454

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc_dec_time.py ..\data\plaintext.txt ..\data\key.txt

Plain Text is:  Welcome to data security and privacy 2019.
Key is:  810a39660fa4507c3cf8e9bab91c6d5b9b4be79aa456930613fb0768324e5027
IV is:  0288e6e120bcbb7df6406f0676e0a735


Ciper is:  2541460d2e0f315cf50c81496523417ba840e5861d7659b13b473e07ffa5242a3711cd8c01ba7d364b0595f2172487df
Encryption Running Time:  0.0001880999999999966

Plaintext is:  Welcome to data security and privacy 2019.
Decryption Running Time:  0.00019720000000000154

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 2 AES\aes_m13523468\src>python enc_dec_time.py ..\data\plaintext.txt ..\data\key.txt

Plain Text is:  Welcome to data security and privacy 2019.
Key is:  810a39660fa4507c3cf8e9bab91c6d5b9b4be79aa456930613fb0768324e5027
IV is:  d6acc5b911ebf8294408c350f8f6110b


Ciper is:  89fe9e839581718eb462d20ef50171129ef6808ca8c99332fb39b5d36d01a8db68034f99e358548bdb360d09375d2db9
Encryption Running Time:  0.00021440000000000348

Plaintext is:  Welcome to data security and privacy 2019.
Decryption Running Time:  0.00022190000000000404
```

It can be observed that the running time for both encryption and decryption is approximately 0.0002 seconds.

4. References

1. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
2. https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm
3. https://pypi.org/project/pycrypto/
4. https://www.dlitz.net/software/pycrypto/api/current/
5. https://docs.python.org/3/library/functions.html#hex