# Unit 8: Report: Cryptography Programming Exercise

## I**ntroduction**

In this unit, the application of cryptographic techniques in Python will be done to enhance data security through the encryption and decryption of text files using an AES encryption algorithm.

## Code Explanation:

### Generating an Encryption Key

We used the Fernet class from the cryptography library to generate a secure encryption key. The key itself was saved for reuse in a file called secret.key so that encryption and decryption can make use of it. In this way, saving the generated key allows both encryption and decryption to use the same key; hence, the integrity and confidentiality of the data are maintained.



Figure 1 : Generating an Encryption Key

### Loading the Encryption Key

A function was then implemented that imported the key that was previously generated from the secret.key file. This is significant to ensure that in decryption processes of whatever data are encrypted with this key, the same key accesses information encrypted through many sessions.

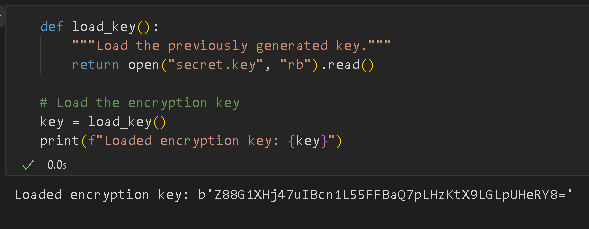


Figure 2: Loading the Encryption Key

### Encrypting a File

The main functionality of the program is to encrypt a text file. The function encrypt\_file reads the content of an input file specified by 'plain\_text.txt', encrypts the content with the loaded encryption key, and then writes out the encrypted data into a new file named 'encrypted\_text.txt'. One example here used a text file containing plain text given below.

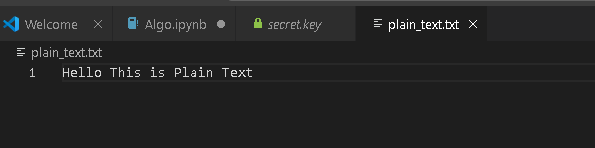


Figure 3: Plain Text File

This step uses the AES encryption algorithm provided by the Fernet class, which ensures that the data is securely encrypted.

### Usage and Execution

It can be observed that after declaring the functions, the program defines the input file to be encrypted and the name of the output file where the encrypted data should be stored.

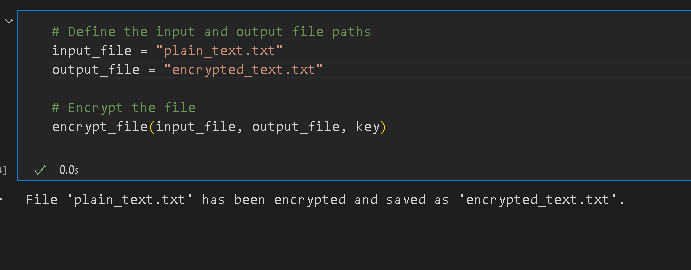


Figure 4: Encrypting File

A call to the encryption function thereafter ensures that the text coming from the source file is safely turned into an unreadable form in this output file, which forms one of the most important features in the AES encryption.

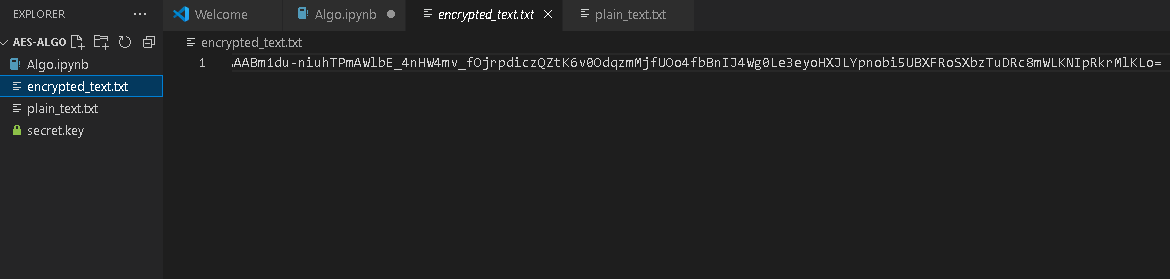


Figure 5: Output: Cipher Text

## Why did you select the algorithm you chose?

With the strong security and wide industry adaptation, AES has been chosen for this exercise (Dang & Vo, 2019). AES is a symmetric encryption algorithm very effective in encrypting bulk data. It supports several key sizes-compared to other encryption algorithms-such as 128, 192, and 256 bits (Kumar et al., 2020). This allows a wide choice based on the intended security level of use. AES is also that it is highly and, importantly, very well-documented with analyses and reviews done by experts in cryptography. As such, it is also more reliable to resist cryptographic attacks. What's more, the cryptography library of Python provides a very easy-to-use interface for implementing AES encryption.

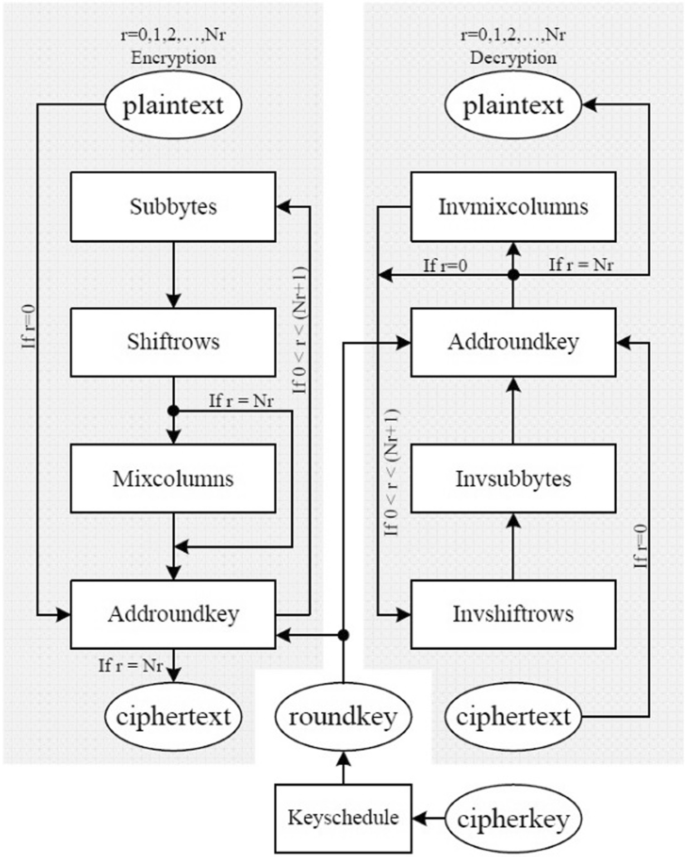


Figure 6: Architecture for AES encryption and decryption: Source: (Equihua et al., 2021)

## Would it meet the GDPR regulations? Justify your answer.

Yes, encryption through AES would meet the GDPR regulations on data protection. According to Caruccio et al. (2020), GDPR indicates that personal data must be processed in a manner that ensures security, partly through encryption, to guard against subjects having unauthorised access to the data and even causing a breach thereby. AES is considered an encryption standard that is secure, and hence, it is commonly used in sensitive information to comply with GDPR ((Bitar & Jakobsson, 2017)

). The major aspects of AES, if implemented accordingly, are packaged in the very fabric of security where encrypted data will not be readable without the key. AES-256 can be used for areas where higher security is required since it provides very strong resistance against all brute-force and other cryptographic attacks. This is why the protection of personal data with encryption methods like AES constitutes a valid response to the GDPR emphasis on security by using appropriate technical measures.

## References

Equihua, C., Anides, E., García, J. L., Vázquez, E., Sánchez, G., Avalos, J. G., & Sánchez, G. (2021). A low-cost and highly compact FPGA-based encryption/decryption architecture for AES algorithm. *IEEE Latin America Transactions*, *19*(9), 1443-1450.

Caruccio, L., Desiato, D., Polese, G., & Tortora, G. (2020). GDPR compliant information confidentiality preservation in big data processing. *IEEE Access*, *8*, 205034-205050.

Bitar, H., & Jakobsson, B. (2017). Gdpr: Securing personal data in compliance with new eu-regulations.

Dang, T. N., & Vo, H. M. (2019, February). Advanced AES algorithm using dynamic key in the internet of things system. In *2019 IEEE 4th international conference on computer and communication systems (ICCCS)* (pp. 682-686). IEEE.

Kumar, K., Ramkumar, K. R., & Kaur, A. (2020, June). A design implementation and comparative analysis of advanced encryption standard (AES) algorithm on FPGA. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 182-185). IEEE.