Advanced Encryption Standard

By Ben Davenport

Introduction:

The Advanced Encryption Standard (AES) is a method for encrypting and decrypting sensitive electronic data that is approved by the Federal Information Processing Standards (FIPS). In short, encryption converts data to an incomprehensible form (ciphertext) and decryption converts the ciphertext back into its original form (plaintext). AES replaced the Data Encryption Standard (DES) in 2001 and has been widely used ever since in areas such as banking, commerce, and communication.

Literature Review:

The first piece of literature, which is a publication from the Federal Information Processing Standard, lays out the nationwide implementation of AES, starting out by letting the reader know why something like AES should be implemented. Sensitive information is incorporated in a wide range of industries where financial data, personal information, and national security secrets could all be jeopardized, causing massive harm to an individual or entity. Encryption algorithms, like AES, can help ensure that only authorized individuals can have access to this sensitive information.

The document also highlights the role of encryption standards, which ensure that encryption algorithms are trustworthy and secure. In order for there to be no weak links in a chain of communication, all parts need to follow the same standard of encryption. The article also lays out background information on symmetric key encryption, including the concepts of keys, plaintext, and ciphertext. In short, symmetric key encryption is described as a type of encryption algorithm that uses the same secret key for both the encryption and decryption of messages. It further says that this type of key encryption is fast and does not require high processing power compared to other algorithms.

The FIPS literature then provides a detailed technical explanation of the AES algorithm, describing the key schedule, encryption rounds, and decryption rounds that are used. The key schedule is the method used to split the secret key into a group of subkeys that are utilized in each encryption and decryption round. The encryption and decryption rounds are the main components of the AES algorithm. The rounds involve a set of operations that convert plaintext into ciphertext and vice versa, such as substitution, permutation, and mixing. The document states that the number of rounds used in the AES algorithm depends on the key length, with longer keys requiring more rounds for greater security.

In addition to the key schedule, encryption rounds, and decryption rounds, the FIPS publication also describes the various modes of operation that can be used with AES. These modes determine how the plaintext is divided into blocks and how the ciphertext is generated. The three main modes of operation described in the document are: Electronic codebook (ECB), Cipher block chaining (CBC), and Counter (CTR). During ECB mode, each block of plaintext is encrypted separately using the same secret key. Although this mode is simple, it is vulnerable to certain attacks such as pattern attacks. CBC mode has each block of plaintext XORed with the previous ciphertext block before encryption begins. This provides greater security than ECB and ensures that identical plaintext segments produce different ciphertext blocks. In CTR mode, a unique counter value is used for each block of plaintext and the counter value is encrypted with the key to produce the keystream. The keystream is then XORed with the plaintext to produce the final ciphertext block. While the literature does provide a detailed explanation of the various modes of use, they don’t outright determine which mode is the best. Pulling in some more outside expertise on the matter, The 10th Conference for Informatics and Information Technology published a paper comparing the modes in terms of level of security [2]. They concluded that out of the three, CTR is the most secure because of its unique counter value and XORed keystream with plaintext.

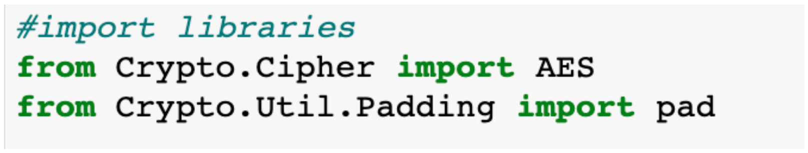
The original FIPS article concludes by discussing potential implementation issues and considerations. The first main concern is key management. The literature emphasizes the importance of key management and stresses the need for proper key generation, storage, and distribution procedures to be established to prevent access to the secret keys. Another important consideration mentioned is the selection of appropriate modes of operation for different applications. Highlighted before, CTR is the most secure, but that is not to say using the other modes aren't valuable. Programmers need to take into account data size, system performance, and security requirements. Lastly, the article mentions the need for continuous monitoring and updating of the AES implementation. Vulnerabilities and threats are constantly changing and the AES usage should be updated regularly to address new weaknesses in the implementation as they emerge.

Methodology/Results:

The program I wrote uses the Python library *Crypto Cipher* that contains algorithms for protecting and encrypting data (imports shown in Figure 1). The program also includes the Crypto Util Padding package that includes methods to take data that may not be a multiple of the block size for a cipher and extend it so that it is (many block cipher modes require the data being encrypted to be an exact multiple of the block size) [3]. In order to use the imported libraries, I first had to install pycryptodome using the command “!pip install pycryptodome”. The written program follows the library's online documentation [4] and is a simple encryption using

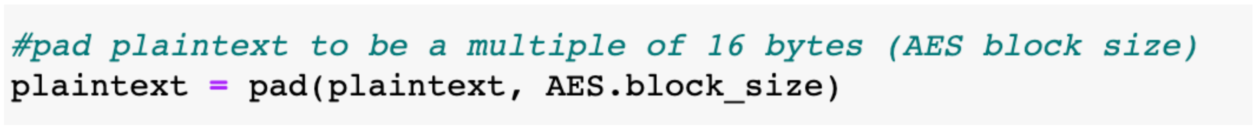
AES that takes a 16 character word for the secret key and a 16 byte word for the plaintext (which is 23 bytes long before padding). After padding it is transformed to be 32 bytes long (two AES block length) which is done in Figure 2. A new AES cipher object is created using a method from the Crypto Cipher package, which takes the key (defined above) and whichever AES cipher mode the user wants to use. In this case, I went with the most secure method, Counter, at the expense of computing power since it was such a small implementation. To encrypt the plaintext, I simply used the cipher.encrypt method, passing through the plaintext as an argument and assigning it to the variable ciphertext. To compare the results, I printed both the plaintext and the ciphertext on top of another to see the before and after encryption.

Every time the program is run, it outputs a different ciphertext that is non legible to the reader, as shown in Figure 3. A good way to check if the cipher is correctly representing the plaintext is to use one of the crypto cipher libraries decryption methods. If done correctly, it will reverse the encryption and return the value of the original plaintext. According to the decryption documentation [5], the recipient can obtain the original message using the same key and the incoming triple in the format (nonce, ciphertext, tag). My program is strictly an encryption algorithm, so decryption is not included, however an example would be to create a new cipher object and pass those as the methods: cipher = AES.new(key, AES.MODE_CTR, nonce=nonce).



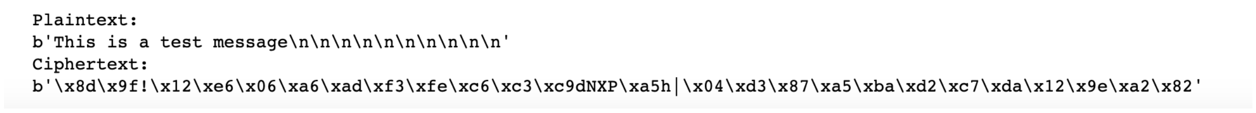
```python
#import libraries
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
```

Figure 1



```python
#pad plaintext to be a multiple of 16 bytes (AES block size)
plaintext = pad(plaintext, AES.block_size)
```

Figure 2



```
Plaintext:
b'This is a test message\n\n\n\n\n\n\n\n\n\n'
Ciphertext:
b'\x8d\x9f!\x12\xe6\x06\xa6\xad\xf3\xfe\xc6\xc3\xc9dNXP\xa5h|\x04\xd3\x87\xa5\xba\xd2\xc7\xda\x12\x9e\xa2\x82'
```

Figure 3

Conclusion:

All in all, the Advanced Encryption Standard (AES) is a widely used method for encrypting sensitive electronic data that was approved for use in 2001. AES uses symmetric key encryption and involves a key schedule, encryption rounds, and decryption rounds. The Federal Information Processing Standard that approved AES discusses the various modes of operation that can be used with AES, including Electronic codebook (ECB), Cipher block chaining (CBC), and Counter (CTR). Out of these, CTR is considered the most secure. The FIPS literature also discusses proper key management, selection of appropriate modes of operation, and continuous monitoring and updating of the AES implementation. Nowadays, simple code cipher libraries can be imported that do most of the heavy lifting in terms of implementation. An example was provided to how one might utilize those packages to encrypt plaintext.

Bibliography:

[1] National Institute of Standards and Technology. (2001). Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[2] Gogova, E., & Mileva, A. (2012). Comparison of AES Modes of Operation. Proceedings of the 10th Conference for Informatics and Information Technology (CIIT). https://doi.org/10.1145/2440940.2441001

[3] Legrand, C. (2018). PyCryptodome Documentation. https://pycryptodome.readthedocs.io/en/latest/src/util/util-padding.html

[4] Legrand, C. (2018). PyCryptodome Documentation. https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

https://www.nist.gov/publications/advanced-encryption-standard-aes