A General Overview of Advanced Encryption Standard (AES)

Arian Eskandarinejad
Independent Researcher
Melbourne, Australia
arianeskandari58@gmail.com
publication date: 12/01/2025

***Abstract*—Advanced Encryption Standard is an encryption block cipher algorithm that is a substitute for DES. It can be implemented as AES-128, 192, or 256, though the block cipher size is constantly 128 bits. This report evaluates a brief background of the formation of AES, a mathematical demonstration of the AES encryption and decryption algorithm along with a short demonstration of the Python code. Furthermore, we discuss the possible cyber attacks against AES and its future substitutes.**

Keywords—Cryptography, Matrix, AES, Rjindael Algorithm, Encryption, Decryption, NIST, CBC.

In 1997, NIST (National Institute of Standards and Technology) commenced a standardization process, aiming to find a substitute for DES (Data Encryption Standard) which requested different candidates worldwide to propose their encryption system. Finally, a new standard named "Rijndael", designed by two Belgian Cryptographers, Joan Daemen and Vincent Rijmen was accepted as the new standard (Simmons, Britannica). This was a very efficient algorithm as it could be used with keys with various lengths, as opposed to DES which was limited to 56-byte key length. NIST later named this algorithm as AES.

I. AES FEATURES

A. Security

AES is extremely resilient to cipher-breaking and this is because the algorithm can use various key lengths i.e. 128, 192, and 256 bits while the same fixed length is applied for both input (plaintext) and output (ciphertext) which is 128 bits. On the other hand, the main mode of operation that AES uses is Cipher-Block Chaining (CBC) which does not show the patterns in a sequence of blocks of encryption. The first plaintext block is XORed with an IV (Initialization Vector), as the Round-0 key, and then each plaintext block value (in the form of a matrix) is XORed with the previous ciphertext block. This chain approach adds a significant level of security to this standard. (Intro to Computer Security, P69)

B. Efficiency

One of the main aspects of efficiency in AES which makes it distinctive is that it is considerably less memory-consuming compared to DES. This can be due to several reasons;

Firstly, as AES uses larger blocks, the need for rapid key scheduling would be reduced. In other words, DES needs to derive keys (based on a pseudo-randomization process) in each round, however in AES, based on the key size corresponding to the number of rounds, 11, 13, or 15 keys are generated all at once, and then saved in the memory.

Secondly, DES uses a Feistel network which is less secure and slower than SPN (Substitution-Permutation Networks) in AES. In Feistel, the input is divided into 2 blocks, and in each round, the round function is run on half of the data to be encrypted and the output is the same size as the input. But in SPN, the input is divided into multiple blocks, and before/after substation (using S-box) and permutation (rows 2-4 of the matrix), the key addition can occur. (Figures 1 and 2)

Thirdly, AES is agile and has a simple and well-defined structure. There are a defined number of rounds with sequential operations (i.e. SubBytes, ShiftRows, MixColumns, AddRoundkey). On the other hand, AES is adaptable to various modes of operations including ECB, GCM, CBC, etc.

Thirdly, AES encryption can be parallelized, meaning that since the encryption for each block is independent, even the encryption process can be done at the same time as decryption. The flexibility allows AES to perform better and consume less memory than DES in modern hardware and software compartments.

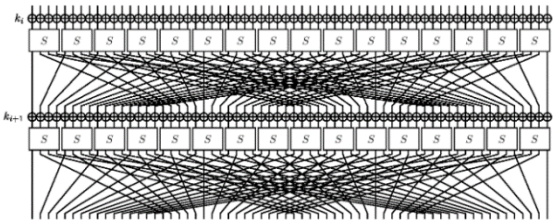

Figure 2 - SPN

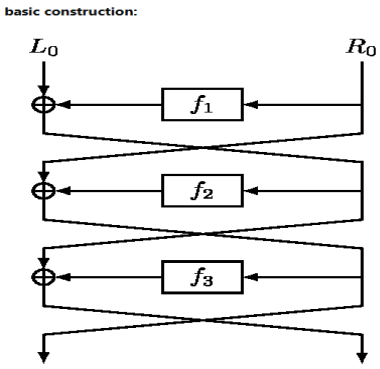

Figure 1 - DES Feistel

II. ENCRYPTION PROCESS

A. Key Generation

The key generation is based on on Pseudo-random Number generator in which a seed is entered as an input so that a reproducible stream of random numbers gets created. The resultant complex would be a key block for round 0 in AES.

On the other hand, it is important to note that in some cases, if the plaintext for the key value is already known, The key can be encrypted with an advanced hash function (using salt value)

B. Data type

This part might be different depending on the business rules or protocol. However, usually, the input (both round 0 key and initial plaintext) can be written in hex, though to complete the operations (e.g. XOR and multiplication), the hex needs to be converted to a binary digit by digit, like the example below:

2F → 2_{binary}: 0010 & F_{binary} = 1111

Hence, 2F in binary is 0010 1111

Remember that the whole hex value (e.g. for a plaintext) 128 bits (16 bytes) is divided into 4 blocks. Each block is 4 bytes and represented as a column in a 4x4 matrix.

C. Key Schedule

When we have the key for round 0, we need to divide the key into 4 segments as w[i] in which i is {1, 2, 3, 4}

[W[0], W[1]. W[2], W[4]]

Each W[i] represents a column in a 4x4 matrix representation. Afterwards, for the third column (e.g. for round 0, W[2]), we use the transformation function g(w[2]) which has the following steps:

1. Circular byte shift to the left e.g. is w[2] = {B7, 5A, 2F, 67}, after left shift, w[2] = {5A, 2F, 67, B7}
2. S-box substitution: use the first digit as a row and the second digit as a column to find the corresponding value in S-box.
3. Adding round constant (Figure 3)

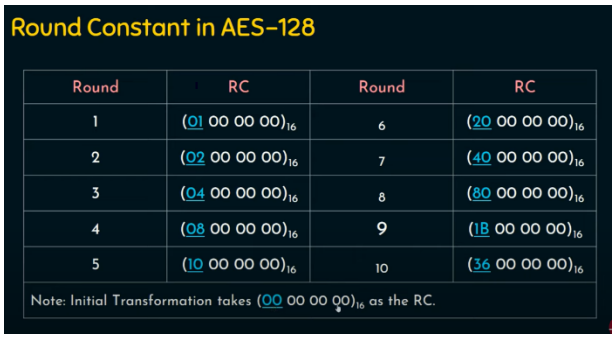

Figure 3 - Round constant table

D. ***SubBytes Step***

In this step, each byte in the matrix is replaced by a byte in the pre-arranged S-Box. The way to perform this is to take the first digit of the hex number as the row and the second digit as the column with S-box (e.g. for 6A, the row is 6 and the column is A, the corresponding substitute value in S-box is 02).

E. ShiftRows Step

In this step, rows 2 to 4 are cyclically shifted left increasingly. In more detail, in this permutational approach, row 1 is shifted by 0, row 2 by 1, row 3 by 2, and row 4 by 3. Please see the example below:

$$A = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \rightarrow \begin{bmatrix} a & b & c & d \\ f & g & h & e \\ k & l & i & j \\ p & m & n & p \end{bmatrix}$$

F. ***MixColumns Step***

Now, we need to multiply a constant matrix derived from the following Galois field formula:

$$x^8 + x^4 + x^3 + x + 1$$

The constant 4x4 matrix has the values 02, 03, 01, 01 and has been permutated/shifted right in rows 2 to 4, increased by 1, hence the resultant matrix is:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

The resultant block (4x4 matrix) will go under the same algorithm again for the next block till we reach the end. Please note that in the last round, the mix columns step is omitted to ensure the decryption is symmetrical.

Below, you can see the diagram of AES algorithms (Figure 4)

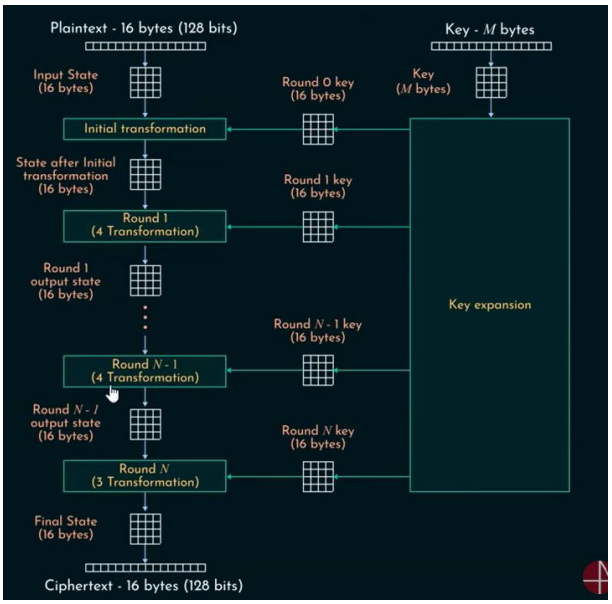

Figure 4 - Structure of AES Algorithm

III. AES IMPLEMENTATION WITH PYTHON

As in real-case scenarios regarding cryptography and encryption algorithms, large inputs and outputs are communicated, apart from using the Crypto library, we have used file management functions to create a file consisting of the ciphertext.

```
From Crypto.Random import get_random_bytes
from Crypto.Protocol.KDF import PBKDF2

From Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpaid

Key_Salt = get_random_bytes(32)
print(Key_Salt)

Key_Plaintext = "I Love Cryptography"

key = PBKDF2(Key_Plaintext, Key_Salt, dkLen=32)

print(key)

Message = b"What is going on?"

ciphertext = AES.new(key, AES.MODE_CBC)

ciphered_data = ciphertext.encrypt(pad(message, AES.block_size))

With open('encrypted.bin', 'wb') as f:
    f.write(ciphertext.iv)
    f.write(ciphered_data)

With open('encrypted.bin', 'RB') as f:
    iv = f.read(16)
    decrypt_data = f.read()
```

```
cipher = AES.new(key, AES.MODE_CBC, iv=iv)
original = unpad(cipher.decrypt(decrypt_data), AES.block_size)
print(original)

With open('key.bin', 'wb') as f:
    f.write(key)
```

In the above code:

We randomly generate 32 byte salt value and then use it to encrypt round 0 key plaintext, using an advanced hash function called PBKDF2.
Afterward, we create a cipher object with the key and put the mode to CBC (it is recommended to use IV as an argument as well).
Finally, we encrypt the padded ciphertext and then write it to a file. In order to decrypt it, we create a new object and unpad and decrypt the cipher.

IV. ATTACKS ON AES

A. *Side-channel Attacks*

Though AES is algorithmically secure and powerful, side-channel attacks can be used for any flaws in computer protocol, mistakes in computation, etc. One of the most common types of this attack is a cache attack, in which the attacker monitors the operation of AES T-table entry (used for optimizing the AES algorithm). Another common type is a timing attack that monitors the data transmission to/from the CPU.

B. *Insecure Modes of Operation*

As mentioned, the standard recommended mode of operation is CBC. In case other modes like ECB (Electronic Codebook) are used, two identical plaintexts would create the same ciphertext which would be a critical issue. This happens because, in ECB, each block is encrypted independently (unlike the key schedule in CBC)

C. *Padding Oracle Attack*

This cryptographic attack is based on trial and error with an algorithmic approach. Suppose that when decrypting AES, the application notices an incorrect padding and raises an exception error. If the attacker gets access to the encrypted message and is aware of the error, he can trial and error for each byte (from right) of the IV by sending a modified ciphertext and doing an XOR operation with the cipher block to eventually find the IV (when no more error is being raised). In summary, the attacker might have the ability to find out the plaintext without even knowing the encryption key.

The most common method to prevent this attack is to use the extension GCM (Galois/Counter Mode) in which Galois field GF (2^{128}) is used to create authentication tag T during encryption.

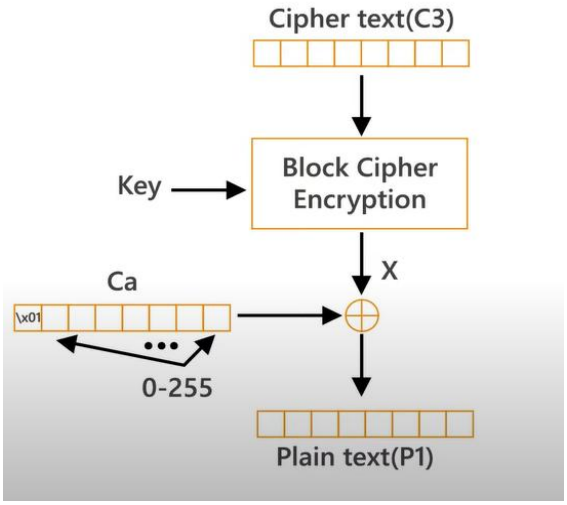

Figure 5 - Simplified Padding Oracle Attack

V. FUTURE ALTERNATIVES FOR AES

Due to the rise of big data and the importance of data security, AES might not satisfy security concerns under some specific circumstances. Some of the main proposed substitutes for AES are:

A. Quantum Cryptography

As the name suggests, the method of cryptography uses the concepts in quantum mechanics to perform cryptographic tasks. The most common application in Quantum Cryptography is Quantum Key Distribution (QKD).
In QKD, a random key is generated for two parties which prevents any eavesdropping or interception, because the attacker would not be able to replicate the quantum state of the key or compromise the concept of superposition (quantum particles can be in multiple positions at the same time) or fundamental theories in quantum physics. Conclusively, any unauthorized interception would raise an alert in the session/system for the parties.

One of the other main advantages of QC is the ability to generate much larger key sizes (more compared to AES-256), which makes it even more secure and reliable.

B. Lightweight Cryptography

The most well-known family of lightweight cryptography is ASCON which has been selected by NIST in 2023 for future use cases.
ASCON is more efficient because firstly it already has integrated authentication, using authenticated encryption with associated data (AEAD). Secondly, ASCON is much more secure against side-channel attacks (One of the reasons is for using a low-degree S-box). Finally, ASCON uses less energy and is computationally less expensive due to its simpler design.

CONCLUSION

In this age of technology, the secure management of data for any business or organization is vital. AES has been proven to be one of the most secure and reliable cryptographic approaches, being flexible and adaptable in different frameworks.

While a detailed mathematical and computational vision is behind AES, the design is relatively simple but capable of handling large data and doing the encryption in a fairly short time.
Some rivals would likely challenge or even substitute AES due to its minor vulnerabilities or areas of improvement, but AES is still useful and performs much better than past algorithms like DES.

REFERENCES

[1] Chirag Bhalodia. (2020, 15 Jul). Key Expansion in AES | Round Constant and g function in Key Expansion in AES [Video]. YouTube. Retrieved from https://www.youtube.com/watch?app=desktop&v=14SAn4IEAHk

[2] Goodrich, M. T., Tamasia, R. (2011). Introduction to Computer Security (International ed). Pearson.

[3] Neso Academy. (2023, 22 Aug). Introduction to Advanced Encryption Standard (AES) [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=3MPkc-PFSRI

[4] NeuralNine. (2022, 4 Jan). Professional Data Encryption in Python [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=gyPuAJfOnGk

[5] Secure Code Warrior (2018, 31 May). Padding Oracle Attack | Explainer Video | Secure Code Warrior [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=lkPBTJ3yiCI

[6] Tobias, E. (2021, 8 Sep). ECB vs. CBC – Pros and Cons of These Block Cipher Modes. Ubiqsecurity. Retrieved from https://www.ubiqsecurity.com/ecb-vs-cbc-block-cipher-mode-differences/#:~:text=As%20discussed%20above%2C%20ECB%20mode,and%20should%20never%20be%20used.

[7] Wikipedia contributors. (2024, 17 Nov). Ascon (cipher). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Ascon_(cipher)

[8] Wikipedia contributors. (2024, 16 Dec). Pseudorandom number generator. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Pseudorandom_number_generator

[9] Wikipedia contributors. (2024, 20 Dec). Quantum Key Distribution. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Quantum_key_distribution

[10] Wikipedia contributors. (2024, 24 Dec). Side-channel attack. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Side-channel_attack