|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Cryptography**

**“Advanced Encryption Standard (AES)”**

Abstract

The abstract is a precise summary of the whole report. Its function is to preview the contents of your report so that the reader can judge whether it is worth their while to read the whole report.

**By:**

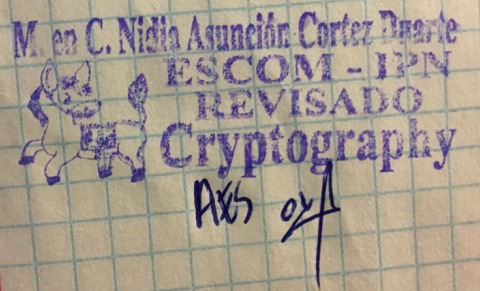
**Garduño Velázquez Alan**

**Romero Gamarra Joel Mauricio**

Professor:

MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

November 2017



**Index**

Contenido

[Introduction: 1](#_Toc499491999)

[Literature review: 3](#_Toc499492000)

[Software (libraries, packages, tools): 3](#_Toc499492001)

[Procedure: 4](#_Toc499492002)

[Results 4](#_Toc499492003)

[Discussion: 4](#_Toc499492004)

[Conclusions: 4](#_Toc499492005)

[References: 5](#_Toc499492006)

[Code 5](#_Toc499492007)

# Introduction:

AES Cipher is a symmetric block cipher capable of encrypt blocks of 128 bits (16 bytes or characters), it has a security of 128 bits because the key to encrypt/decrypt the information has 128 bits of length. This cipher, receive arrays of 16 bytes (it doesn’t matter if we are talking about plaintext, images, wav files, etc.). [1]

It is formed by 4 different functions in regular rounds (9) and 3 of those 4 functions in the last round, on Figure 1 is shown the general process of AES.

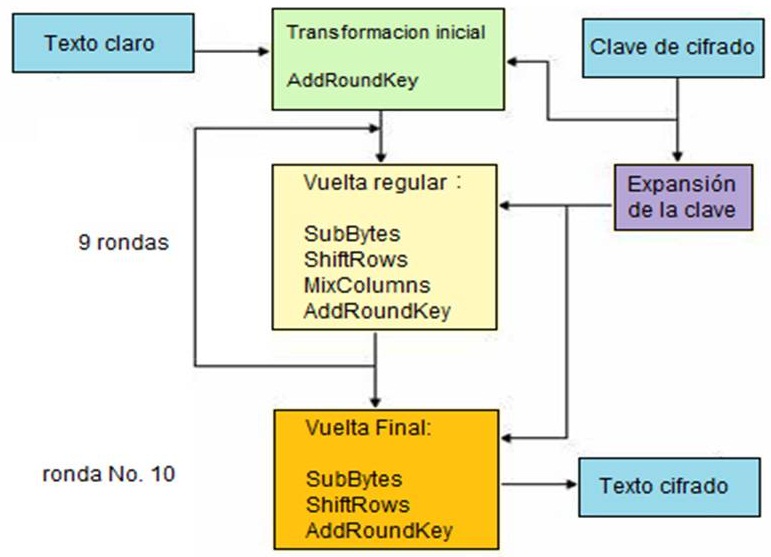


Figure 1. General Diagram of AES

In this cipher, the plaintext is represented by a matrix as such as the key (it has its own process of expansion to select a new key for the next rounds).

As we can observe, the 4 different functions are:

* **AddRoundKey:** XOR operation byte by byte between key and plaintext.
* **SubBytes:** Substitute each byte of the matrix following the S-box.
* **ShiftRows:** Make a left-rotation of n – bytes (with n from 0 to number of rows – 1).
* **MixColumns:** Multiply the matrix obtained by the first 3 functions with the key in GF (28).

Although these 4 functions seems to be very simple, just 3 of them really are, because MixColumns function is very hard if we do it manually, because a multiplication of 2 matrixes in GF (28) is not too simple. This procedure will be explained better on “Literature Review” section.

This cipher works with bytes (I already mentioned that) written in hexadeximal, that’s why GF (28), each byte is composed by 8 bits, but we need only 4 bits to represent from 0 to F and then, we have to have 2 *characters* on each position of the matrix.

On Figure 2, we can observe AddRoundKey function.

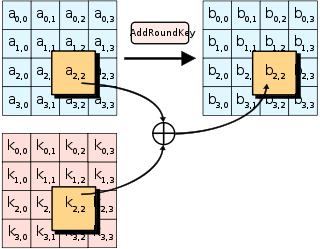


Figure 2. AddRoundKey Function

As you can see, we made and XOR between a2,2 and k2,2 to obtain b2,2. The next function (SubBytes) is shown on Figure 3.

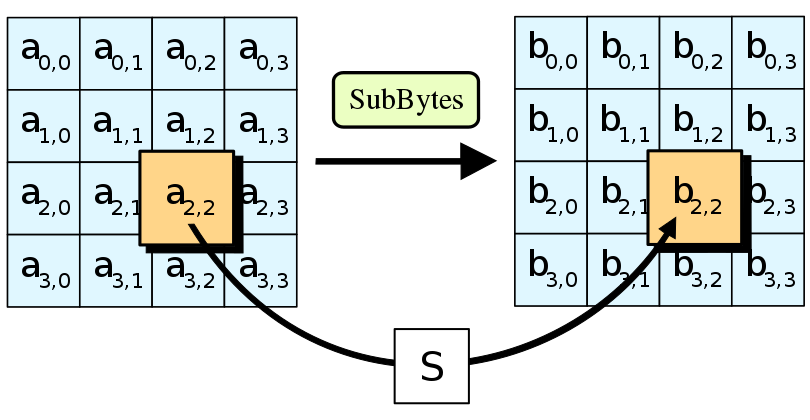


Figure 3. SubBytes Function

On Figure 4, is shown ShiftRows Function.

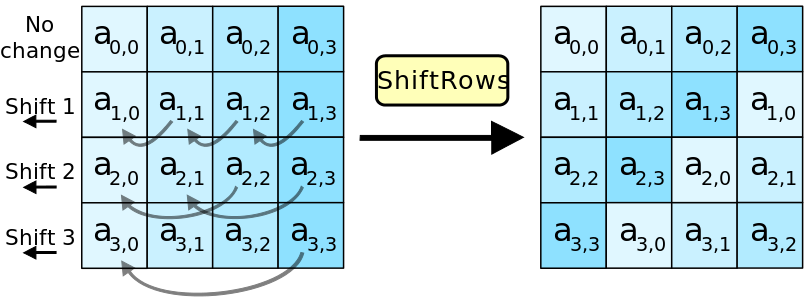


Figure 4. ShiftRows Function

The last function, MixColums, is shown on Figure 5.

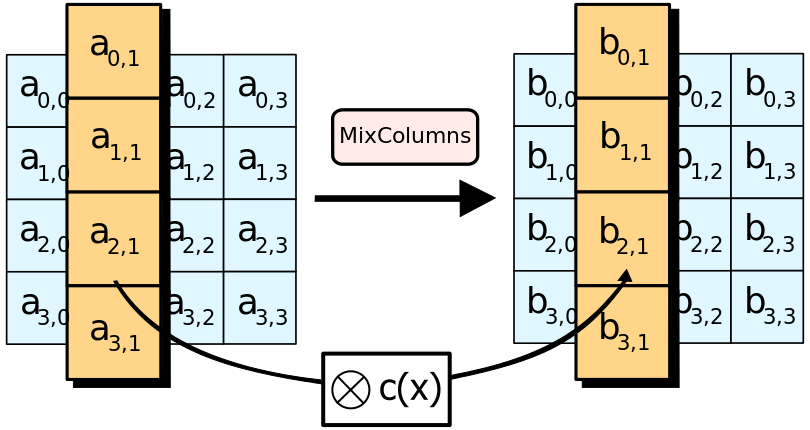


Figure 5. MixColumns Function

On the next section, I will explain the idea of MixColumns function supported by multiplication in GF (28).

# Literature review:

The literature needs to provide an understanding of the conceptual and theoretical and mathematical background, context and justification of your work.

You should include diagrams, formulas, algortithms, …

should be referenced using

# Software (libraries, packages, tools):

**Packages [2]:**

* Crypto: A collection of cryptographic modules implementing various algorithms and protocols.

**Sub - packages [2]:**

* Cipher: Used for the following objects and/or methods
  + AES [3]: Cipher object
  + Encrypt: Method used to encrypt data with key and parameters set at initialization
  + Decrypt: Method used to decrypt data with key and parameters set at initialization

**Tools:**

* Sublime Text 3 [4]
* Python 3.6 [5]

# Procedure:

\* Flowchart / block diagram  
\* Add details (step-by-step) of your procedure in such a way that anyone else could repeat the experiment.

# Results

\* This section should include any data tables, observations, images.   
\* All tables, graphs and charts should be labeled appropriately.

This section describes but does not explain your results

Since you are presenting your results, not the figures which represent the results, you should ensure you refer explicitly to your results and not just to your data figures (graphs, tables). As you describe particular results in the text of your results section, make sure you refer to the corresponding figure in brackets after you have mentioned the results. The figures should be inserted into the text as soon as possible after you mention them.

# Discussion:

Your discussion section has two fundamental aims:

to interpret and explain the results of your study,

to explore the significance of your study’s findings. [qualify and explore](https://unilearning.uow.edu.au/report/2bvi.html) the theoretical importance/significance of your results.

The discussion is also the place in a report where any qualifications or reservations you have about the research should be aired.

# Conclusions:

\* List one thing you learned and describe how it applies to a real-life situation.  
\*Discuss possible errors that could have occurred in the collection of the data (experimental errors)

\*How generally do your results apply?

\*Were there any defects in your experimental design or procedure?

# References:

**[1]** Federal Information Processing Standards, “Advanced Encryption Standard (AES)”, November 2001. [Online]. Available: [https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/ fips-197.pdf](https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/%20fips-197.pdf)

**[2]** EpyDoc, “API Documentation - Package Crypto”, May 2012. [Online]. Available: https://www. dlitz.net/software/pycrypto/api/2.6/

**[3]** EpyDoc, “API Documentation – Class AESCipher”, May 2012. [Online]. Available: <https://www.dlitz.net/software/pycrypto/api/2.6/Crypto.Cipher.AES.AESCipher-class.html>

**[4]** Sublime HQ Pty Ltd, “Sublime Text – Download”, November 2017. [Online]. Available: <https://www.sublimetext.com/3>

**[5]** Python Software Foundation, “python”, October 2017. [Online]. Available: https://www.python. org/downloads/

# Code

**AES.py**

**import** os

**from** Crypto.Cipher **import** AES

**from** Crypto.Util **import** Counter

**def** main ():

os.system ("cls")

#We define the functions (modes of operation) inside a dictionary

modos\_operacion = {1: ECB, 2: CBC, 3: CFB, 4: OFB, 5: CTR}

#Menu to the user

original = input ("\nEnter the name of the original image: ")

cipher = input ("\nEnter the name of the encrypted image: ")

option = int (input ("\nSelect an option\n\n1.Enrypt\n2.Decrypt\n\n"))

**print** ("\n\nWhich mode of operation do you want?\n\n")

**print** ("1. Electronic Codebook (ECB)")

**print** ("2. Cipher Block Chaining (CBC)")

**print** ("3. Cipher Feedback (CFB)")

**print** ("4. Output Feedback (OFB)")

**print** ("5. Counter (CTR)")

mode = int (input ("\n"))

#Calling the selected mode of operation

modos\_operacion [mode] (original, cipher, option)

#Electronic Codebook

**def** ECB (original, ciphered, option):

#Asking for the key to the user (16 bytes)

key = bytes (input ('Introduce the key: '), 'utf-8')

#Creating a new AES cipher

cipher = AES.new (key, AES.MODE\_ECB)

#Opening both files

original\_file = open (original, "rb")

encrypted\_file = open (ciphered, "wb")

#We copy the entire head of the image

data = original\_file.read (54)

encrypted\_file.write (data)

#Obtaining the size of the image

original\_file.seek (34)

size = int.from\_bytes (original\_file.read (4), byteorder = 'little')

#We move to the start of the real image to encrypt it

original\_file.seek (54)

i = 0

#Encrypt

**if** option == 1:

**while** (i < size):

#Reading 16 bytes to encrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.encrypt (pixels)

#Writing encrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was encrypted correctly using Electronic Codebook Mode")

#Decrypt

**elif** option == 2:

**while** (i < size):

#Reading 16 bytes to decrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.decrypt (pixels)

#Writing decrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was decrypted correctly using Electronic Codebook Mode")

original\_file.close ()

encrypted\_file.close ()

#Cipher Block Chaining

**def** CBC (original, ciphered, option):

#Asking for the key to the user (16 bytes)

key = bytes (input ('Introduce the key: '), 'utf-8')

#Asking for the initialization vector to the user (16 bytes)

IV = bytes (input ('Introduce the initialization vector: '), 'utf-8')

#Creating a new AES cipher

cipher = AES.new (key, AES.MODE\_CBC, IV)

#Opening both files

original\_file = open (original, "rb")

encrypted\_file = open (ciphered, "wb")

#We copy the entire head of the image

data = original\_file.read (54)

encrypted\_file.write (data)

#Obtaining the size of the image

original\_file.seek (34)

size = int.from\_bytes (original\_file.read (4), byteorder = 'little')

#We move to the start of the real image to encrypt it

original\_file.seek (54)

i = 0

#Encrypt

**if** option == 1:

**while** (i < size):

#Reading 16 bytes to encrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.encrypt (pixels)

#Writing encrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was encrypted correctly using Cipher Block Chaining Mode")

#Decrypt

**elif** option == 2:

**while** (i < size):

#Reading 16 bytes to decrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.decrypt (pixels)

#Writing decrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was decrypted correctly using Cipher Block Chaining Mode")

original\_file.close ()

encrypted\_file.close ()

#Cipher Feedback

**def** CFB (original, ciphered, option):

#Asking for the key to the user (16 bytes)

key = bytes (input ('Introduce the key: '), 'utf-8')

#Asking for the initialization vector to the user (16 bytes)

IV = bytes (input ('Introduce the initialization vector: '), 'utf-8')

#Creating a new AES cipher

cipher = AES.new (key, AES.MODE\_CFB, IV)

#Opening both files

original\_file = open (original, "rb")

encrypted\_file = open (ciphered, "wb")

#We copy the entire head of the image

data = original\_file.read (54)

encrypted\_file.write (data)

#Obtaining the size of the image

original\_file.seek (34)

size = int.from\_bytes (original\_file.read (4), byteorder = 'little')

#We move to the start of the real image to encrypt it

original\_file.seek (54)

i = 0

#Encrypt

**if** option == 1:

**while** (i < size):

#Reading 16 bytes to encrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.encrypt (pixels)

#Writing encrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was encrypted correctly using Cipher Feedback Mode")

#Decrypt

**elif** option == 2:

**while** (i < size):

#Reading 16 bytes to decrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.decrypt (pixels)

#Writing decrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was decrypted correctly using Cipher Feedback Mode")

original\_file.close ()

encrypted\_file.close ()

#Output Feedback

**def** OFB(original,ciphered,option):

#Asking for the key to the user (16 bytes)

key = bytes (input ('Introduce the key: '), 'utf-8')

#Asking for the initialization vector to the user (16 bytes)

IV = bytes (input ('Introduce the initialization vector: '), 'utf-8')

#Creating a new AES cipher

cipher = AES.new (key, AES.MODE\_OFB, IV)

#Opening both files

original\_file = open (original, "rb")

encrypted\_file = open (ciphered, "wb")

#We copy the entire head of the image

data = original\_file.read (54)

encrypted\_file.write (data)

#Obtaining the size of the image

original\_file.seek (34)

size = int.from\_bytes (original\_file.read (4), byteorder = 'little')

#We move to the start of the real image to encrypt it

original\_file.seek (54)

i = 0

#Encrypt

**if** option == 1:

**while** (i < size):

#Reading 16 bytes to encrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.encrypt (pixels)

#Writing encrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was encrypted correctly using Output Feedback Mode")

#Decrypt

**elif** option == 2:

**while** (i < size):

#Reading 16 bytes to decrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.decrypt (pixels)

#Writing decrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was decrypted correctly using Output Feedback Mode")

original\_file.close ()

encrypted\_file.close ()

#Counter

**def** CTR(original,ciphered,option):

#Asking for the key to the user (16 bytes)

key = bytes (input ('Introduce the key: '), 'utf-8')

#Creating a new AES cipher

ctr = Counter.new (128)

cipher = AES.new (key, AES.MODE\_CTR, counter = ctr)

#Opening both files

original\_file = open (original, "rb")

encrypted\_file = open (ciphered, "wb")

#We copy the entire head of the image

data = original\_file.read (54)

encrypted\_file.write (data)

#Obtaining the size of the image

original\_file.seek (34)

size = int.from\_bytes (original\_file.read (4), byteorder = 'little')

#We move to the start of the real image to encrypt it

original\_file.seek (54)

i = 0

#Encrypt

**if** option == 1:

**while** (i < size):

#Reading 16 bytes to encrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.encrypt (pixels)

#Writing encrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was encrypted correctly using Counter Mode")

#Decrypt

**elif** option == 2:

**while** (i < size):

#Reading 16 bytes to decrypt it using AES cipher

pixels = original\_file.read (16)

#Encrypting 16 bytes readed

encrypted\_pixels = cipher.decrypt (pixels)

#Writing decrypted pixels

encrypted\_file.write (encrypted\_pixels)

#Updating the counter

i = i + 16

**print** ("\n\n", original, "was decrypted correctly using Counter Mode")

original\_file.close ()

encrypted\_file.close ()

#Main function

main ()