

# ECRYPT – AES SOFTWARE IMPLEMENTATION

Jakub Szpila 283644

## Algorytm description:

The algorithm works by repeating a series of simple bit and positional manipulations a set number of times. The amount of repetitions is dictated by the length of the provided key (128bit key – 10 rounds, 196 – 12 rounds and 256 – 14 rounds).

All operations are done by grouping the data into bytes, and said bytes into 4x4 blocks with 128bit blocks.

In each round 4 operations are made on a block :

subBytes : each byte is substituted with another one from a pre-calculated lookup table, called SBox. The SBox is designed in such a way that there are no pairs (A->B, B->A) and no byte has all bit flip.

ShiftRows : In every block the rows are shifted (in a cycle) by a set number of spaces to the left.

1<sup>st</sup> row has no shifts

2<sup>nd</sup> row is shifted by 1 to the left :

A	B	C	D
---	---	---	---

Is shifted to

B	C	D	A
---	---	---	---

3<sup>rd</sup> row is shifted by 2 spots

4<sup>th</sup> row is shifted by 3 spots (so 1 spot to the right)

Mix columns : Each column is mixed up by using a matrix multiplication with a polynomial. The matrix looks like this:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

And the whole block is multiplied by it. The multiplication is done in modulo 283, and the addition is done with XOR operation on each byte.

Add round key : The whole block is XOR'ed with the key corresponding to the current round.

All rounds are done using this 4 operations except for the 0<sup>th</sup> one that uses only the add round key function and the last one that doesn't use mix columns function.

For decryption the process is done in reverse, using inversions of described functions.

### **Functional description:**

The implementation works best, and is easiest to test if all input is done by providing the program with inputs in hex format (2 digits per number).

It should look like this :

**b26aeb1874e47ca8358ff22378f09144**

There should be no spaces between the numbers/digits. When the input is done confirm with <enter>

Output is provided the same way, in hex, both for output in blocks (useful for debugging) and in a string.

### **Code structure description:**

The code was split into smaller, easier to work chunks. Each part used in the algorithm (whole system, single block or single byte) was split into different class and all functions related to it were enclosed in corresponding class.

All functions should have some brief description in their code.

Byte class : responsible for byte operations. All functions that happen to single byte (substitution, XORs etc.) were placed there.

Block class : This class was responsible for all block wide operations. Printing, shifting, mixing and similar were written in this class. The class also has some functions related to creation of a block from a string or appropriate array.

All functions related to the block operation have their inversion written there (mix\_columns and inv\_mix\_columns etc.)

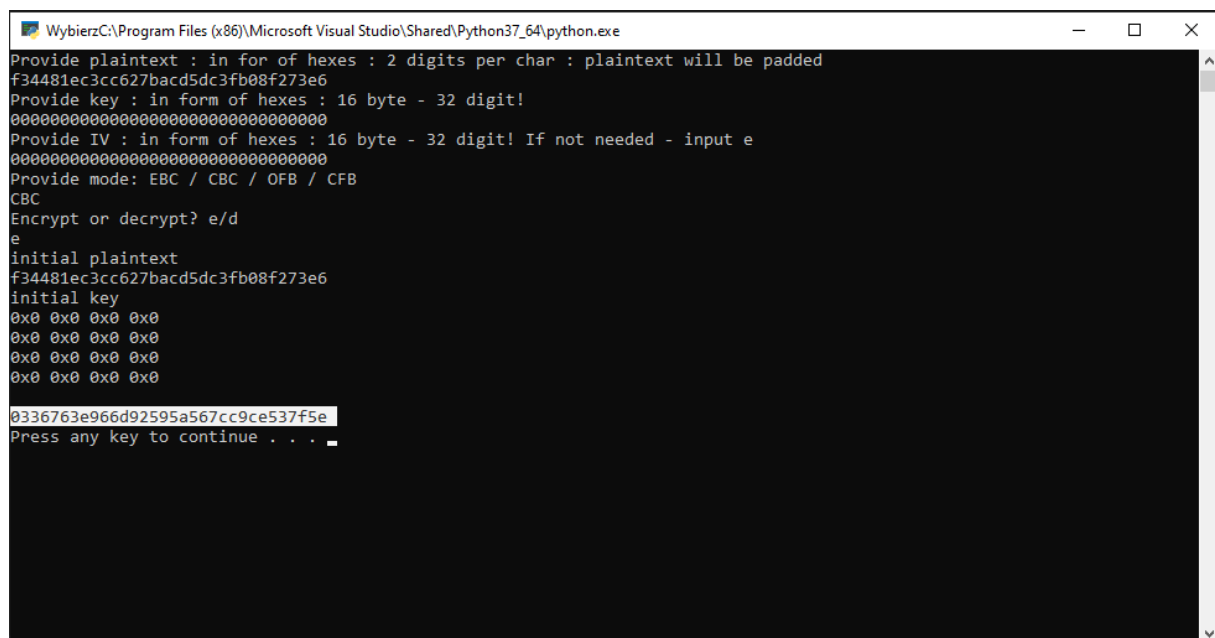
AES\_System class : This was the biggest class. It holds functions and information related to the whole information and process (key and plaintext,

number of rounds, mode of operation etc.). This is also where all encryption/decryption functions (for all supported modes) was implemented as well as key expansion algorithm was implemented.

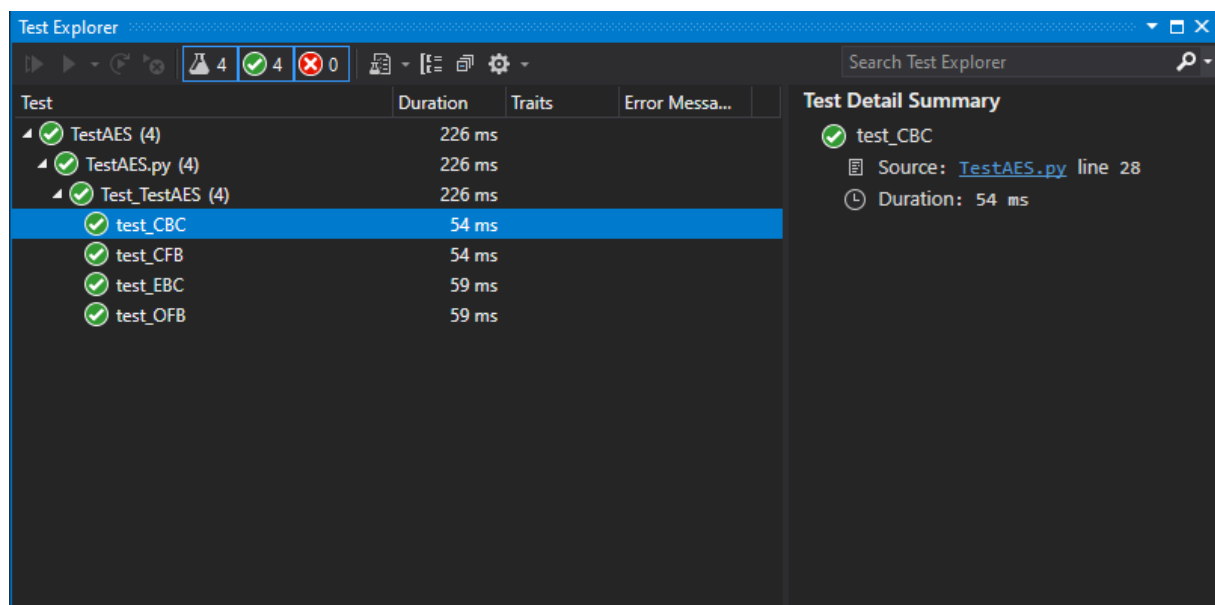
## Tests:

Test data was taken from AES documentation, from <https://csrc.nist.gov/>  
exact link : <https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/aes/AESAVS.pdf>

A group of 5 tests cases was created from provided data, and each method was tested against it. All tests can be viewed and run in test explorer in Visual studio. Below : example of the program in operation and the view of all tests passed (couldn't get the tests themselves to output to console)



```
WybierzC:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe
Provide plaintext : in for of hexes : 2 digits per char : plaintext will be padded
f34481ec3cc627bacd5dc3fb08f273e6
Provide key : in form of hexes : 16 byte - 32 digit!
00000000000000000000000000000000
Provide IV : in form of hexes : 16 byte - 32 digit! If not needed - input e
00000000000000000000000000000000
Provide mode: EBC / CBC / OFB / CFB
CBC
Encrypt or decrypt? e/d
e
initial plaintext
f34481ec3cc627bacd5dc3fb08f273e6
initial key
0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0
0336763e966d92595a567cc9ce537f5e
Press any key to continue . . .
```



Test	Duration	Traits	Error Messa...
TestAES (4)	226 ms		
TestAES.py (4)	226 ms		
Test_TestAES (4)	226 ms		
test_CBC	54 ms		
test_CFB	54 ms		
test_EBC	59 ms		
test_OFB	59 ms		

**Test Detail Summary**  
test\_CBC  
Source: TestAES.py line 28  
Duration: 54 ms