AES Algorithm Walkthrough

A step by step guide to the principles used in AES.

**Requirements:**

* A plaintext message
* A passkey for encryption

**Maths Required:**

1. XOR operator: A **⊕** B

Convert numbers to 8 bit binary.

Add digits corresponding to same power of 2.

Finally take remainder when divided by 2 to get each digit

Eg: 78 **⊕** 43

78 = “0b01001110”

43 = “0b00101011”

78 **⊕** 43 = “0b01100101” = 101

1. Dot Product: (A.B)

Convert numbers into 8 bit binary.

From these binary forms,

We create polynomials in x where the coefficient of x^n

is the digit corresponding to 2^n in the binary form.

Eg:

78 = “0b01001110”

-> x+x^2+x^3+x^6

43 = “0b00101011”

1+x+x^3+x^5

Now, multiply these polynomials.

-> x^11 + x^9 + x^8 + 2x^7 + 3x^6 + x^5 + 2x^4+2x^3

+ 2x^2 + x

Reduce the polynomial to degree 8 ( to convert back to 8 bit binary) by finding remainder when divided by the standard polynomial of AES: x^8 +x^4 + x^3 + x +1

-> remainder = x^7 + 2x^6 – x^4 + x^2 - x – 1

This is converted back to binary by first taking each coefficient’s remainder by 2 , then making nth binary digit the new coefficient of x^n

-> Final result: “0b10010111” = 151

**ENCRYPTION:**

1. Accept an input message and convert all characters into standard 8 bit ASCII values

If the number of characters is not a multiple of 16, extend message length to the same.

This method is called “salting” in cryptography.

The method used in the python code is adding spaces

Eg: Message :

“Sooper secret message”

L = [83, 111, 111, 112, 101, 114, 32, 115, 101, 99, 114, 101, 116, 32, 109, 101,115, 115, 97, 103, 101, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32] # ord(“ “) = 32

1. Convert this list into a nested list of sublist length 16.

(Reason is to carry out further steps.)

Eg:

[[83, 111, 111, 112, 101, 114, 32, 115, 101, 99, 114, 101, 116, 32, 109, 101], [115, 115, 97, 103, 101, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32]]

1. Accept the key and carry out steps 1 and 2. But this time we only want a 16 byte key, so we convert all integers to 8 bit binary and we xor the corresponding elements of the sublist.

Eg: Key: “Not so secret anymore”

-> [[78, 111, 116, 32, 115, 111, 32, 115, 101, 99, 114, 101, 116, 32, 97, 110], [121, 109, 111, 114, 101, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32]]

-> [['0b01001110', '0b01101111', '0b01110100', '0b00100000', '0b01110011', '0b01101111', '0b00100000', '0b01110011', '0b01100101', '0b01100011', '0b01110010', '0b01100101', '0b01110100', '0b00100000', '0b01100001', '0b01101110'], ['0b01111001', '0b01101101', '0b01101111', '0b01110010', '0b01100101', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000', '0b00100000']]

-> (After xor):

[55, 2, 27, 82, 22, 79, 0, 83, 69, 67, 82, 69, 84, 0, 65, 78]

1. **KEY GENERATION:**

There are 10 rounds in 128 bit(refers to key size) AES encryption. So we generate 10 keys known as “roundkeys”

* + We start with key0 or the given key.
  + We take 4 bytes(1 row) at a time out of 16 bytes and modify them. Store a copy of the 4 byte rows and keep them aside.
  + The first step is to mix up the rows by bringing the last byte to the top. This step is called “rotword”
  + Next, the sbox table is looked up and each byte is substituted according to the same.
  + Finally, the “rcon” (a special predefined list) is looked up according to the round number. This gives us the final piece of the puzzle. The rcon round list (length 4) is generated simply by taking the round value and adding 0s to get 4 bytes.
  + The 3 components formed are:
* The original copy of 4 bytes of the previous key
* The substituted key
* Rcon round list
  + Next we xor the corresponding elements of the 3 lists.
  + This becomes one row of the round key.
  + We club the rows back together to give the full round key.
  + Generate key 1 to key 10 in the same way from key0.

Note: rcon = [0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36]

Eg: Round 4:

1 row: [2,56,78,33]

After rotword: [33,2,56,78]

After substitution: [253,119,7,47]

Rcon list : [8,0,0,0]

After xor: [247,79,73,14]

1. **The Actual Rounds:**

For 9 rounds we repeat the following steps:

1. Adding the round key:

First take the key for the given round (round 1 – key0

Round2 – key 1 …)

Simply xor the corresponding elements of the key and each sublist of 16 bytes from the previous output.

1. S Box (substitution box):

Substitute each and every byte by looking at the sbox table (used in key generation)

Now onwards we deal only with 16 bytes at a time.

1. Shift rows:

First arrange the 16 bytes in a grid vertically.

|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B4 | B8 | B12 |
| B1 | B5 | B9 | B13 |
| B2 | B6 | B10 | B14 |
| B3 | B7 | B11 | B15 |

Row 0

Row 1

Row 2

Row 3

The elements in each row are shifted to the right by a number equal to the row index.

|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B4 | B8 | B12 |
| B13 | B1 | B5 | B9 |
| B10 | B14 | B2 | B6 |
| B7 | B11 | B15 | B3 |

Now the elements are put back together to give the original 16 elements.

[B0,B13,B10,B7,B4,B1,B14,B11,B8,B5,B2,B15,B12,B9,B6,B3]

1. Mix columns:

Arrange all the 16 bytes in the same way as the previous step to give a 4\*4 matrix.

|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B4 | B8 | B12 |
| B1 | B5 | B9 | B13 |
| B2 | B6 | B10 | B14 |
| B3 | B7 | B11 | B15 |

Now we do a matrix multiplication.

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 3 | 1 | 1 |
| 1 | 2 | 3 | 1 |
| 1 | 1 | 2 | 3 |
| 3 | 1 | 1 | 2 |

Standard Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B4 | B8 | B12 |
| B1 | B5 | B9 | B13 |
| B2 | B6 | B10 | B14 |
| B3 | B7 | B11 | B15 |

\*

Each element is obtained from multiplying corresponding elements of a row and a column and then adding them up. This would give a 4\*4 matrix.

Here instead of multiplying we compute the dot product of corresponding elements and instead of adding we xor the values.

Eg: Element in 3rd row, 4th column of final matrix:

[1,1,2,3] \* [B12,B13,B14,B15]

= [(1 . B12) **⊕** (1 . B13) **⊕** (2 . B14) **⊕** (3 . B15)]

[1,1,2,3] \* [46,79,34,83]

= [(1 . 46) **⊕** (1 . 79) **⊕** (2 . 34) **⊕** (3 . 83)]

= [46 **⊕** 79 **⊕** 68 **⊕** 245]

Finally, the resultant matrix is put back together to give the result of the round.

1. Repeat steps c and d for all other s16 byte sublists.
2. For the tenth round, carry out steps a through e, except step d(no mix columns)
3. Add key10 to give the final encrypted message in ASCII.
4. Convert the ascii numbers back into the original characters to give the ciphertext.