Network Security

Programming Exercise Set 2

Project 1 : *Advanced Encryption Standard (AES)*

Manak Bisht 2018340

Nitish Gupta 2018351

Thursday 8th October, 2020

1 System Design

The implementation works with keys of size 128 bit and has 10 rounds. The 128 bit blocks of plaintext are implicitly represented as a 4x4 matrix of bytes, called "state" with each byte stored as a two digit hexadecimal number.

1.1 AES class

1.1.1 constructor AES

"The constructor may be called with or without a key. If the key is not specified, it generates a random key itself using the `generateKey` *function. The created object is always associated with the key at the time of creation. The key cannot be modified after creation. This object may be used for encryption or decryption by calling the* `encrypt` *and* `decrypt` *functions respectively. A*

new object needs to created for working with a different key."

Parameter - `key length or key`

Returns - `Class instance with the associated key`

1.1.2 substituteBytes

"An in-place byte-by-byte substitution which requires first taking an invertible affine transformation and then then finding the multiplicative inverse in the Galois Field (2^8)*."*

Parameter - `state`

Returns - `void`

1.1.3 inverseSubstituteBytes

"An in-place byte-by-byte substitution which requires first taking the inverse of the affine transformation and then finding the multiplicative inverse in the Galois Field (2^8)*."*

Parameter - `state`

Returns - `void`

1.1.4 shiftRows

"Applies circular left shift in-place to the rows of the input matrix. The first, second, third and fourth rows are circularly shifted left by $0, 1, 2, 3$ *positions respectively."*

Parameter - `state`

Returns - `void`

1.1.5 inverseShiftRows

"Applies circular right shift in-place to the rows of the input matrix. The first, second, third and fourth rows are circularly shifted right by $0, 1, 2, 3$ *positions respectively."*

Parameter - `state`

Returns - `void`

1.1.6 mixColumns

"Combines the four bytes of each column of the state using an invertible linear transformation ***`mixColTransformation`****. The arithmetic is performed in Galois Field* (2^8)*."*

Parameter - `state`

Returns - `void`

1.1.7 inverseMixColumns

"Combines the four bytes of each column of the state using ***`inverseMixColTransformation`*** *linear transformation. The arithmetic is performed in Galois Field* (2^8)*."*

Parameter - `state`

Returns - `void`

1.1.8 addRoundKey

"The round key is added by performing a bitwise XOR of each byte of the state with the corresponding byte of the round key. The round key is derived from the main key."

Parameter - `state, round key`

Returns - `void`

1.1.9 nextRoundKey

"Generates the next round key given the previous round key. j is the round number. $RC_{j+1} = 2 \times RC_j$ in Galois Field (2^8) & $RC_1 = 01$. Also, implicitly calls the ***`gFunction`****, a circular left shift of the last column of the previous round key, followed by substitution and XOR with RC_j."*

Parameter - `previous round key,` RC_j

Returns - `void`

1.1.10 generateAllRoundKeys

"From the main key, generates round keys for every round. The key and round keys are associated with the object, stored as instance variables"

Parameter - `none`

Returns - `void`

1.1.11 encrypt

"Performs all encryption rounds on the plaintext. Implicitly calls ***`roundEncrption`*** *ten times."*

Parameter - `plaintext`

Returns - `ciphertext`

1.1.12 decrypt

"Performs all decryption rounds on the ciphertext. Implicitly calls ***`roundDecryption`*** *ten times."*

Parameter - `ciphertext`

Returns - `plaintext`

1.2 GF256 class

AES uses the Galois Field (2^8) for all arithmetic with the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. The class has two static methods `sum` & `product`.

1.2.1 sum

"Addition is simply a bitwise XOR of **`x`** *&* **`y`***."*

Parameter - `x, y`

Returns - `x+y`

1.2.2 product

"Multiplication is performed modulo the irreducible polynomial with the coefficients modulo 2."

Parameter - `x, y`

Returns - `x×y`

2 Examples

Example 1

Key - `F75DA76803CFBDA3ACBC726167A09E83`

Plaintext - `3567B9BBA58B94A9441F3F6E47FD2406`

Ciphertext - `99ECDB57B51FE0FA45BB10A59559CABF`

Example 2

Key - `0617CBF5840AB34724451A83A760490F`

Plaintext - `8B362C8D31CC1DE906EE3A68A5234D3B`

Ciphertext - `4FB939FD3AAA9CE08FCEB959EB5BC1E8`