Relatório Segurança de Sistemas
AES CBC e CTR:
Encriptação e Decriptação

Hugo Neto[1]

[1]Escola Politécnica - Faculdade de Informática (PUCRS)
Porto Alegre – RS – Brazil
Github: https://github.com/hugonn/aes-sha

1. Introdução

O intuito deste relatório é demonstrar,resumidamente, o funcionamento e a implementação de um sistema que encripta/decripta mensagens utilizando a técnica de encriptação AES e demonstrar dois modos de operação , CBC e CTR. A tecnologia escolhida foi a linguagem Python que faz usufruto da biblioteca PyCrypto. O AES é um método de encriptação simétrica que faz o uso de blocos de 16 bytes e um ”bloco falso”(IV/ Vetor de Inicialização) que é utilizado no início do processo de encriptação de cada encriptação dos blocos. Esta técnica é extremamente segura. Tendo em vista que, na força bruta, é muito difícil de quebrar uma cifra que foi encriptada com o método AES.

2. Modos de Operação AES

O AES possui vários modos de operação, mas apenas dois foram explorados nesse sistema:

- CBC: Na encriptação, todos os blocos não cifrados realizam uma operação de XOR com os 16 bytes iniciais do bloco anterior. O primeiro bloco faz esta operação utilizando o IV, enquanto que os blocos subsequentes utilizam os 16 bytes iniciais do bloco antecedente. Isso garante a aleatoriedade e segurança da cifra.

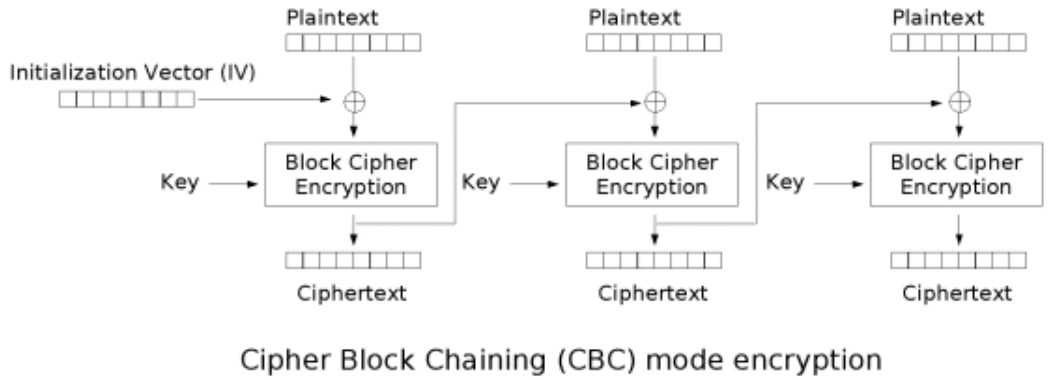

Figura 1. Modo: CBC

- CTR: Esse modo cria uma cifra de fluxo onde, para cada bloco de texto não criptografado, é feito uma operação XOR com um contador que é incrementado antes de cada cifragem nos blocos subsequentes.

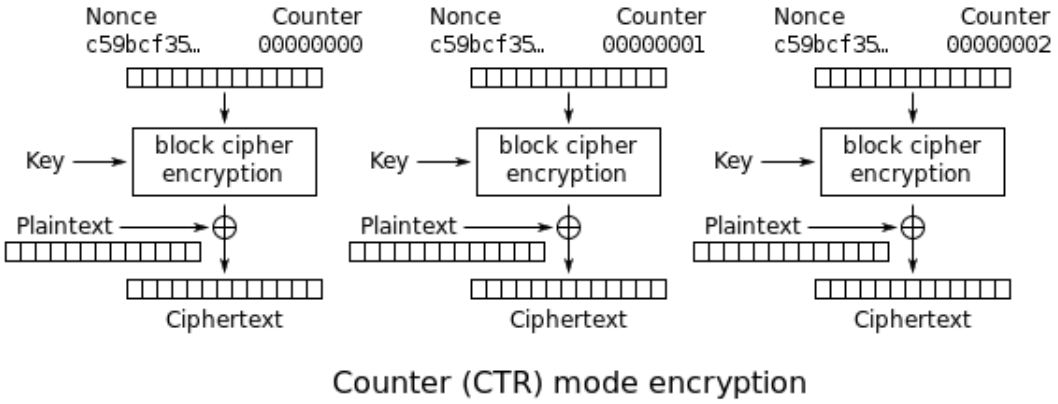

Figura 2. Modo: CTR

3. Implementação

A tecnologia escolhida foi Python em conjunto com a biblioteca PyCrypto. O intuito de utilizar essa tecnologia foi aprender uma linguagem na qual não possuia conhecimento.

As imagens abaixo demonstram partes da implementação do sistema de encriptação/decriptação:

```
def encrypt(key, source, encode=True):
    source = unhexlify(source)
    key = unhexlify(key)
    IV = Random.new().read(16)  # Gera IV
    encryptor = AES.new(key, AES.MODE_CBC, IV) #Encripta
    padding = AES.block_size - len(source) % AES.block_size  # Calcula PAD
    source += bytes([padding]) * padding  # Padding
    data = IV + encryptor.encrypt(source)# IV + Cifra
    return  hexlify(data)
```

Figura 3. Modo: CBC

```
#Arquivo que contem o decriptador AES_hex
def decriptar_hex(cifra,chave):
    cifra = unhexlify(cifra)    #Transforma em bytes a informação que está hexa
    chave = unhexlify(chave)    #Transforma em bytes a informação que está hexa
    IV = cifra[:16]             #Recupera o IV dos 16 primeiros bytes
    aes_obj = AES.new(chave, AES.MODE_CBC, IV) #Utiliza função do PyCrypto
    texto = aes_obj.decrypt(cifra[16:]) #Decripta a mensagem sem o IV
    padding = texto[-1]                 #Retira o pad
    return bytes.decode(texto[:-padding]) #Decodifica os Bytes
```

Figura 4. Modo: CBC

As partes do código mostradas acima, são responsáveis pela encriptação e decriptação, respectivamente, utilizando o método AES com modo de operação CBC. Dentro do problema proposto, foi necessário transformar as cifras/texto claro da base hexadecimal para base decimal e, subsequentemente, criar um IV randômico para encriptar e para decriptar recuperar os primeiros 16 bytes do primeiro bloco da cifra. Foi utilizado PKCS5 para encriptar as cifras no modo CBC.

```
def int_of_string(s):
    return int(binascii.hexlify(s), 16) # Contador aleatorio por conta do CTR

def encrypt_CTR(key, texto):
    texto = unhexlify(texto) #Transforma em bytes a informação que está hexa
    key = unhexlify(key)     #Transforma em bytes a informação que está hexa
    iv = Random.new().read(16)     #Cria o "bloco falso"
    ctr = Counter.new(128, initial_value=int_of_string(iv)) #cria objeto que encripta no modo CTR
    aes = AES.new(key, AES.MODE_CTR, counter=ctr) #encripta cifra
    return hexlify(iv + aes.encrypt(texto))
```

Figura 5. Modo: CTR

```
def int_of_string(s):
    return int(binascii.hexlify(s), 16) #contador aleatorio por conta do CTR


def decrypt_CTR(key, cifra):
    cifra = unhexlify(cifra)    #Transforma em bytes a informação que está hexa
    key = unhexlify(key)        #Transforma em bytes a informação que está hexa
    iv = cifra[:16]             #Cria o "bloco falso"
    ctr = Counter.new(128, initial_value=int_of_string(iv)) #Cria objeto AES em modo CTR
    aes = AES.new(key, AES.MODE_CTR, counter=ctr) # Decripta cifra
    return bytes.decode(aes.decrypt(cifra[16:]))
```

Figura 6. Modo: CTR

As imagens acima demonstram como foi feita a encriptação/decriptação utilizando a técnica AES em modo CTR. O contador randômico é utilizado para encriptar blocos subsequentes. Na decriptação, a cifra é transformada de hexadecimal para decimal e, após isso, é criado o objeto AES que decripta as mensagens propostas no trabalho.

Abaixo, estão os resultados das encriptações e decriptações do AES tanto no modo CTR quando no CBC.

```
Mensagem Encriptada -----> 69dda8455c7dd4254bf353b773304eec0ec7702330098ce7f7520d1cbbb20fc388d1b0adb5054dbd7370849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f
51eeca32eabedd9afa9329
Mensagem Clara ------> CTR mode lets you build a stream cipher from a block cipher.

Mensagem Encriptada -----> 770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa0e311bde9d4e01726d3184c34451
Mensagem Clara ------> Always avoid the two time pad!
```

Figura 7. Resultados Encriptação/Decriptação

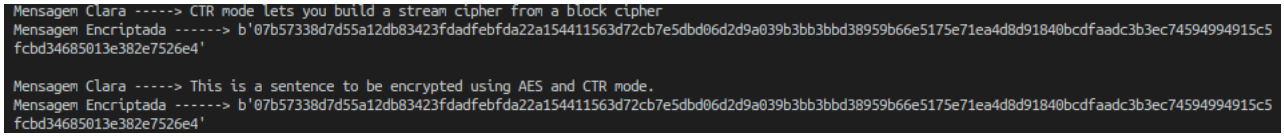

Figura 8. Resultados Encriptação/Decriptação

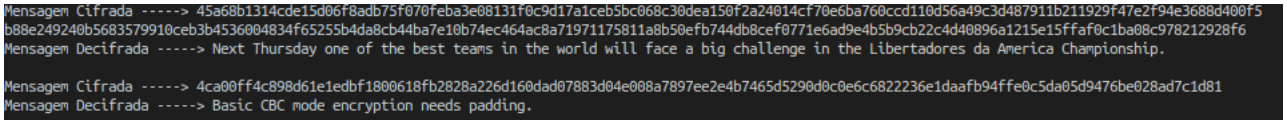

Figura 9. Resultados Encriptação/Decriptação

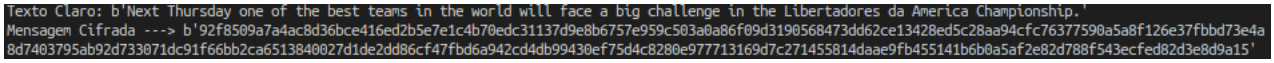

Figura 10. Resultados Encriptação/Decriptação

4. Conclusão

A metodologia de encriptação AES continua sendo muito utilizada nos dias de hoje devido a sua eficiência e segurança dos dados. Pode-se dizer que é impossível quebrar as cifras AES tendo em vista que só consegue-se saber o que tem nas mensagens cifradas caso a chave seja conhecida. Sendo assim, este trabalho foi motivador e foi de suma importância para a aprendizagem desse método de encriptação.

Referências

Morais, J. (2017). Criptografia aes. Último Acesso: 20 Novembro de 2018.