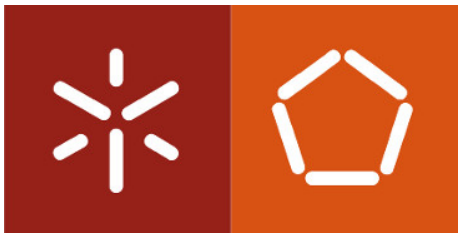
Tecnologia Criptográfica

22 de Novembro de 2020

Trabalho Prático 3

a83899	André Morais
a84485	Tiago Magalhães

Segurança de cifra por blocos do algoritmo AES no modo CBC e ECB



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Introdução **2**

2 Resultados **2**

3 Conclusão e observação de resultados **3**

4 Anexos **4**
4.1 Método ECB 4
4.2 Método CBC 6

1 Introdução

No âmbito da Unidade Curricular de Tecnologia Criptográfica, foi nos proposto que utilizássemos uma cifra por blocos neste caso o algoritmo AES no modo ECB (*Electronic Codebook*) e CBC (*Cipher Block Chaining*) para cifrar uma imagem e verificássemos que mesmo utilizando este tipo de cifra a sua segurança poderia ser quebrada utilizando um modo inseguro.

2 Resultados

A imagem utilizada para a realização do trabalho foi a seguinte:

Figura 1: Imagem .bmp utilizada para cifragem nos diferentes modos do algoritmo AES.

Resultados da imagem cifrada para o modo ECB e CBC:

(a) Imagem cifrada no modo ECB

(b) Imagem cifrada no modo CBC

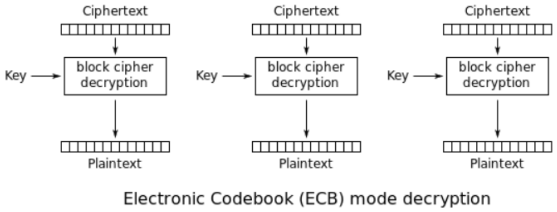
Figura 2: Resultados

3 Conclusão e observação de resultados

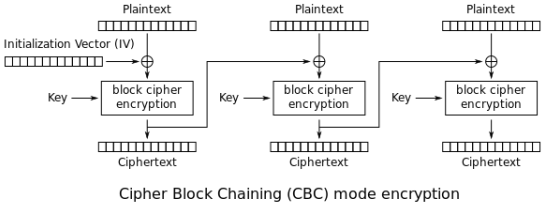
Como podemos ver pelos resultados da aplicação dos modos ECB e CBC do algoritmo AES o mais seguro é o CBC, uma vez que não revela informação acerca da mensagem original, neste caso imagem.

A obtenção destes resultados deve-se ao facto de que no modo ECB, como podemos ver pela **figura 3(a)** cada bloco ser cifrado independentemente dos outros, como a nossa imagem apresenta contornos parecidos, estes blocos ao serem cifrados vão ter um resultado determinístico, isto é, o resultado da cifragem vai ser sempre igual, assim, ao observarmos o resultado iremos encontrar padrões ao da imagem original.

Pelo contrário no modo CBC, a cifragem dos blocos é dependente dos blocos anteriores (**figura 3(b)**), deste modo não iremos obter padrões, apesar desta dependência ao cifrar a imagem iríamos obter sempre o mesmo resultado, mas com a utilização de um vector de inicialização a cifragem passa ser não determinística e o resultado da cifragem ser sempre diferente(não reutilização do vetor de inicialização).



(a) AES no modo ECB



(b) AES no modo CBC

Figura 3: AES nos modos ECB e CBC

4 Anexos

4.1 Método ECB

```
import os
import sys
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.ciphers import Cipher,algorithms,modes

os.system("dd if=./panda.bmp of=panda_enc.bmp bs=1 count=54 conv=notrunc")

# abre imagem bmp que irá ser cifrada
img = open("./panda.bmp","rb")

# guarda em data os bytes da imagem
data = img.read()
img.close()

# gera chave para o algoritmo AES no modo  ECB
key = os.urandom(32)

# seleciona o encription scheme
cipher = Cipher(algorithms.AES(key), modes.ECB())

# retorna instancia encryptor que irá ser usada para cifragem
encryptor = cipher.encryptor()

# seleciona algoritmo de padding
padder = padding.PKCS7(algorithms.AES.block_size).padder()

# adiciona padding ao último bloco de bytes da imagem de modo a ter tamanho
# do bloco do algoritmo AES
padded = padder.update(data)

# finaliza operação
padded += padder.finalize()

# cifra dados
ct=encryptor.update(padded)+encryptor.finalize()

# retorna instancia decryptor que irá ser usada para decifrarmgem
decryptor = cipher.decryptor()

# algoritmo para retirar padding para decifragem
unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
```

```
# decifra imagem
imgdata = decryptor.update(ct)

# retira bytes adicionados à imagem
unpadded = unpadder.update(imgdata) + unpadder.finalize()

# escreve para ficheiro resultado da imagem cifrada
f = open('./panda_enc.bmp', 'ab')
f.write(ct)
f.close()

# escreve para ficheiro resultado da imagem decifrada
f = open('./panda_dec.bmp','wb')
f.write(unpadded)
f.close()
```

4.2 Método CBC

```
import os
import sys
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

os.system("dd if=./panda.bmp of=panda_enc.bmp bs=1 count=54 conv=notrunc")

img = open("./panda.bmp","rb")
data = img.read()
img.close()

key = os.urandom(32)
iv = os.urandom(16)

cipher = Cipher(algorithms.AES(key),modes.CBC(iv))

encryptor = cipher.encryptor()

padder = padding.PKCS7(algorithms.AES.block_size).padder()
padded = padder.update(data)
padded += padder.finalize()

ct=encryptor.update(padded)+encryptor.finalize()

decryptor = cipher.decryptor()

unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
imgdata = decryptor.update(ct)
unpadded = unpadder.update(imgdata) + unpadder.finalize()

f = open('./panda_enc.bmp', 'ab')
f.write(ct)
f.close()

f = open('./panda_dec.bmp','ab')
f.write(unpadded)
f.close()
```

Referências

[1] Biblioteca Python cryptography:
`https://cryptography.io/en/latest/`