Trabalho 2
Advanced Encryption Standard

Davi Jesus de Almeida Paturi, 20/0016784

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0201 - Segurança Computacional

davi.paturi@aluno.unb.br

Resumo. *O Advanced Encryption Standard (AES) é um algoritmo de criptografia amplamente utilizado em todo o mundo para proteger informações confidenciais. Foi adotado como padrão pelo governo dos Estados Unidos em 2001 e é uma escolha comum para a criptografia de dados em várias aplicações, como segurança de rede, armazenamento de dados e comunicações seguras.*

1. Introdução

A criptografia desempenha um papel crítico na segurança da informação no mundo digital. Um dos pilares mais sólidos nesse campo é o Advanced Encryption Standard (AES), um algoritmo de criptografia amplamente adotado e respeitado. O AES é uma conquista notável no campo da criptografia, sendo utilizado para proteger dados confidenciais em várias aplicações, como comunicações seguras, armazenamento de dados e transações financeiras. Neste trabalho, exploraremos os princípios básicos do algoritmo de criptografia AES.

Esse trabalho está dividido em 4 seções. A seção 2 apresenta as ferramentas e os metódos utilizados para o desenvolvimento desse trabalho. Na seção 3 está a implementação de modelos de cifragem e decifragem usando o algoritmo AES. A seção 4 exibe a implementação do modo de operação CTR. Na seção 5 estão expostos os resultados e suas respectivas análises.

2. Metodologia

A implementação dos algoritmos de cifragem e decifragem, bem como o algoritmo de ataque, foram feitos em Python. O código fonte das implementações estão disponíveis em [Github 2023].

3. Algoritimo AES

O algoritmo de criptografia Advanced Encryption Standard (AES) é um método simétrico amplamente usado para proteger a confidencialidade de dados em várias aplicações. Ele opera em blocos de dados de tamanho fixo, geralmente 128 bits, e utiliza chaves de criptografia de diferentes tamanhos (128, 192 ou 256 bits). As etapas de criptografia do AES envolvem uma série de transformações complexas que são executadas em várias rodadas [Wikipédia 2023].

Antes de começar essas etapas, o bloco de dados é convertido em uma matriz bidimensional de 4x4 bytes (uma matriz de estado) e a chave de criptografia é utilizada para geração de chaves de rodada. Após a geração desse bloco de dados, as etapas explicadas a seguir são executadas em ordem.

3.1. Adição de chave de rodada inicial

O primeiro passo envolve a combinação do bloco de dados com uma chave de rodada derivada da chave de criptografia original. Isso é feito executando uma operação XOR entre o bloco de dados e a chave de rodada. Cada byte do bloco de dados é combinado com o byte correspondente na chave de rodada. O código dessa etapa segue abaixo.

```
def AddRoundKey(block, key):
  return [v ^ key[i] for i, v in enumerate(block)]
```

3.2. Rounds de transformação

Cada rodada consiste em várias etapas de processamento, incluindo uma que depende da própria chave de criptografia. Cada rodada consiste nas seguintes etapas:

3.2.1. Substituição de bytes

Cada byte no bloco de dados é substituído por um novo byte usando uma tabela de substituição chamada SBox. Isso introduz não linearidade e confusão nos dados.

```
def SubBytes(block):
  return [Sbox[v] for v in block]
```

3.2.2. Permutação de linhas

Os bytes nas linhas do bloco de dados são permutados. Os bytes na primeira linha não são alterados, o segundo byte da primeira linha é movido para a segunda linha, o terceiro byte da primeira linha é movido para a terceira linha e o quarto byte da primeira linha é movido para a última linha. Isso ajuda a dispersar os bytes no bloco de dados.

```
def ShiftRows(block):
  for i in range(4):
    block[i*4 : i*4+4] = Rotate(block[i*4 : i*4+4], i)
  return block
```

3.2.3. Mistura de colunas

As colunas do bloco de dados são misturadas através de operações matemáticas. Isso envolve operações de multiplicação e adição em campo de Galois (um corpo finito). Essa etapa ajuda a difundir os bytes e aumenta a confusão nos dados.

```
def MixSingleColumn(c):
  t = c[0] ^ c[1] ^ c[2] ^ c[3]
  u = c[0]
  c[0] ^= GaloisMult(c[0] ^ c[1]) ^ t
  c[1] ^= GaloisMult(c[1] ^ c[2]) ^ t
  c[2] ^= GaloisMult(c[2] ^ c[3]) ^ t
  c[3] ^= GaloisMult(c[3] ^ u) ^ t
  return c

def MixColumn(block):
```

```
  l1, l2, l3, l4 = [], [], [], []
  for i in range(4):
    column = MixSingleColumn([block[i], block[i+4], block[i+8], block[i
    +12]])
    l1.append(column[0])
    l2.append(column[1])
    l3.append(column[2])
    l4.append(column[3])
  return (l1 + l2 + l3 + l4)
```

3.2.4. Adição de chave de rodada

A chave de rodada é combinada com o bloco de dados por meio da operação XOR. Cada rodada usa uma chave de rodada diferente, que é derivada da chave de criptografia original. O código dessa etapa é o mesmo da etapa 3.1.

3.3. Rodada final

A última rodada não inclui a etapa 3.2.3. Em vez disso, ela segue as etapas 3.2.1, 3.2.2 e 3.2.4 antes de prosseguir para a saída criptografada. O resultado final é a saída criptografada, que consiste em um bloco de dados de 128 bits.

3.4. Descriptografia

O processo de descriptografia do algoritmo AES envolve a aplicação das mesmas etapas em ordem reversa.

- Inversa da mistura de colunas.
- Inversa da permutação de linhas.
- Inversa da substituição de bytes.
- Adição de chave de rodada.

4. Modo de operação CTR

O modo CTR requer uma chave de criptografia, um contador inicial (nonce) e um valor inicial (IV - Initial Vector). O nonce é um valor que geralmente é único para cada mensagem ou sessão de criptografia. O IV é um valor aleatório que nunca deve ser reutilizado com a mesma chave. Para efeitos didáticos, o valor de IV começará com o valor zero nos nossos experimentos.

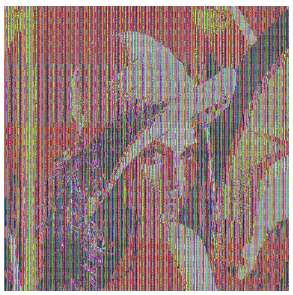
O modo CTR usa um contador que começa com o valor do IV. Em cada bloco, o contador é incrementado para gerar o próximo valor. Por exemplo, o primeiro bloco é cifrado com o IV, o segundo bloco com o IV + 1, o terceiro com o IV + 2 e assim por diante. O código da implementação desse modo pode ser encontrado no arquivo ”CTR.py”em [Github 2023].

5. Resultados

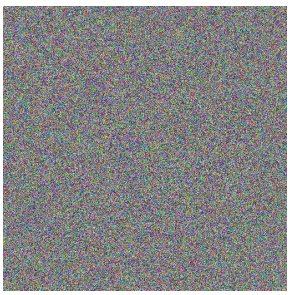
Para analisarmos o efeito da criptografia AES, iremos aplicar o algoritmo AES com modo de operação CTR na Figura 1, com quatro diferentes números de rodadas e faremos uma comparação dos resultados.

Figura 1: Imagem original sem criptografia.

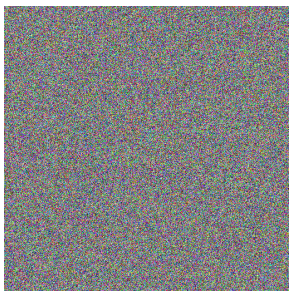
Comparando a Figura 1 com a Figura 2a, é perceptível que apenas uma rodada não é suficiente para criptografar uma imagem de maneira eficiente, já que boa parte dos contornos e das cores da imagem ainda são visíveis. A partir de cinco rodadas (Figura 2b), o resultado fica indistinguível da imagem original, sendo assim, uma maneira eficiente de criptografar uma imagem. O processo de criptografia com nove rodadas (Figura 2c) e com treze rodadas (Figura 2d) também geram um resultado indistinguível, fornecendo uma maior segurança acerca do conteúdo da imagem cifrada.



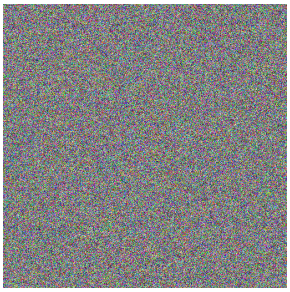
(a) Imagem criptografada com 1 rodada.



(b) Imagem criptografada com 5 rodadas.



(c) Imagem criptografada com 9 rodadas.



(d) Imagem criptografada com 13 rodadas.

Figura 2: Imagens criptografadas com diferentes níveis de rodadas.

Referências

[Github 2023] Github (2023). Projeto 2. https://github.com/davipatury/SC-T2. [Online].

[Wikipédia 2023] Wikipédia (2023). Advanced encryption standard – wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard. [Online; acessado em 28 de outubro de 2023].