Trabalho de Implementação
Cifra de bloco e modo de operação CTR

Nicole de Oliveira Sena *

Universidade de Brasília, 28 de agosto de 2024

Resumo

Este relatório descreve a implementação da cifra de bloco AES e o modo de operação CTR (contador), tendo três partes: implementação da cifra, do modo de operação e teste.

Palavras-chave: Criptografia · AES · Cifra de Bloco · Advanced Encryption Standard · Rijndael

1 Introdução

O padrão de criptografia avançada - advanced encryption standard (AES), é uma especificação para a criptografia de dados eletrônicos um subconjunto da cifra de bloco. O algoritmo descrito pelo AES é um algoritmo de chave simétrica, o que significa que a mesma chave é usada para criptografar e descriptografar os dados. [1]

Uma cifra de bloco é um algoritmo determinístico que opera sobre agrupamentos de bits de tamanho fixo, chamados de *blocos*. Uma cifra de bloco consiste de dois algoritmos emparelhados, um para encriptação, E, e outro para a decriptação, D.[2]

O AES é uma variante do Rijndael, com um tamanho de bloco fixo de 128 bits e um tamanho de chave de 128, 192 ou 256 bits.[1] Neste trabalho, usaremos uma *chave fixa de 128 bits*. Esta implementação permite especificar o número de rodadas que você deseja executar.

2 Estrutura do AES

O AES opera em um arranjo de bytes de ordem principal de linha e de coluna 4×4, denominada estado.[1] O tamanho de chave usado para uma cifra AES especifica o número de rodadas de transformação que convertem a entrada, para 128 bits, serão 11 rodadas.

Para o funcionamento do AES são necessários alguns dados como aS–Box(tabela desubstituição estática), o estado (que é o bloco de dados de entrada sendo modificado pelastransformações do algoritmo), a chave, e a chave de expansão (uma versão modificada da chave). O AES é dividido em dois módulos, sendo um para cifragem e um para decifragem.[3]

O módulo para cifragem conta com quatro transformações:

- SubBytes: Transformação que substitui os bytes do estado por bytes da S–Box;
- ShiftRows: rotaciona ciclicamente as linhas da matriz 4x4 do estado para a esquerda, para a segunda em 1 casa, para a terceira em 2 casas e para a quarta 3 casas;
- MixColumns: esta operação transforma os dados das colunas do estado multiplicando por um polinômio irredutível fixado;
- AddRoundKey: “mistura” as colunas com uma das chaves geradas na rotina de expansão. Uma operação de XOR bit a bit do estado com a chave.

No módulo de decifragem existem transformações equivalentes contrárias, como a SubBytes inversa (que utiliza umaS–Box inversa), a ShiftRows inversa, a MixColumns inversa e a AddRoundKey inversa. Cada uma dessas faz a operação inversa de sua representante no módulo de cifragem.[3]

3 Etapas

Para uma encriptação de 128 bits o algoritmo AES realiza as seguintes etapas:

- Primeira rodada:
  - AddRoundKey
- Rodadas 2 à 9:
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- Rodada final (totalizando 11 rodadas):
  - SubBytes
  - ShiftRows
  - AddRoundKey

3.1 Modos de Operação

Algumas técnicas de criptografia simétrica cifram os arquivos em blocos de tamanho fixo para que seja possível ter acesso a partes deles sem a necessidade de cifra-los ou decifrá-los por completo. Os blocos são cifrados utilizando cifras de bloco que subdividem esses blocos em blocos menores processando-os individualmente. [4]

A forma como esses blocos menores são processados durante a cifragem deve seguir regras específicas para que sejam mantidas as propriedades da segurança da informação. Essas regras constituem o que é chamado de modos de operação. [4]

O modo CTR consiste na utilização de um contador que é incrementado a cada novo bloco processado. Esse contador, denominado **nonce**, é submetido ao processo de cifragem para depois ser realizado um *XOR* com o texto a ser cifrado. A etapa final de *XOR* resulta no texto cifrado. Esse processo pode ser visto na figura 1.

*190114860@aluno.unb.br

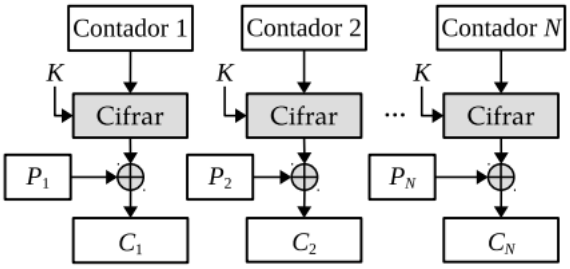

Figura 1: *Modo CTR*

3.2 Código

Como parte do processo, transforma cada estado de entrada em um novo valor usando uma matriz S-box, retirada do site Medium [5]. Nesse caso, a tabela S-box é uma matriz de 16x16 que utiliza cada valor de entrada, onde os primeiros quatro bits são usados para definir a linha da tabela, e os próximos quatro bits definem a coluna. Por exemplo, se o byte de entrada for CF, então a saída será 8A. A S-box inversa (também retirada de Medium [5]) reverte o processo da S-box, de modo que DF se refere a CF.

Detalharemos a seguir algumas funções correspondentes às etapas já mencionadas. Além delas, temos funções de cifragem e decifragem, operações relacionadas ao modo CTR e manipulação de mensagens que não serão detalhadas.

3.2.1 SubBytes

A função subBytes realiza uma substituição de bytes em um estado, utilizando uma tabela de substituição conhecida como S-box. A função mapeia cada byte do estado para um novo valor de acordo com essa tabela.

```
        def sub_bytes(state):
            return [s_box[b] for b in state]


        def inversa_sub_bytes(state):
            return [inversa_s_box[b] for b in state]
```

3.2.2 Shift Row

Neste processo, a seguinte transformação é aplicada:

- A primeira linha permanece inalterada.
- A segunda linha sofre uma rotação circular de um byte para a esquerda.
- A terceira linha é deslocada dois bytes para a esquerda.
- A quarta linha é deslocada três bytes para a esquerda.

```
        def shift_rows(state):
            return [
                state[0], state[5], state[10], state[15],
                state[4], state[9], state[14], state[3],
                state[8], state[13], state[2], state[7],
                state[12], state[1], state[6], state[11]
            ]


        def inv_shift_rows(state):
            return [
                state[0], state[13], state[10], state[7],
                state[4], state[1], state[14], state[11],
                state[8], state[5], state[2], state[15],
                state[12], state[9], state[6], state[3]
            ]
```

3.2.3 Mix Columns

Nessa etapa, cada coluna é processada uma de cada vez, e cada byte da coluna é transformado em um novo valor com base nos quatro bytes da coluna (onde utilizamos a multiplicação no campo de Galois).

```
         def galois_mult(a, b):
            if a == 0 or b == 0:
                return 0
            if a == 1 or b == 1:
                return a * b
```

```
            return E[(L[a] + L[b]) % 0xff]


        def mix_columns(state, matriz):
            for i in range(4):
                column = state[i * 4: i * 4 + 4]
                state[i * 4] = galois_mult(matriz[0][0], column[0]) ^ galois_mult(matriz
                    [0][1], column[1]) ^ galois_mult(matriz[0][2], column[2]) ^
                    galois_mult(matriz[0][3], column[3])
                state[i * 4 + 1] = galois_mult(matriz[1][0], column[0]) ^ galois_mult(
                    matriz[1][1], column[1]) ^ galois_mult(matriz[1][2], column[2]) ^
                    galois_mult(matriz[1][3], column[3])
                state[i * 4 + 2] = galois_mult(matriz[2][0], column[0]) ^ galois_mult(
                    matriz[2][1], column[1]) ^ galois_mult(matriz[2][2], column[2]) ^
                    galois_mult(matriz[2][3], column[3])
                state[i * 4 + 3] = galois_mult(matriz[3][0], column[0]) ^ galois_mult(
                    matriz[3][1], column[1]) ^ galois_mult(matriz[3][2], column[2]) ^
                    galois_mult(matriz[3][3], column[3])
            return state
```

3.2.4 Add Round Key

Aqui, implementamos uma operação XOR entre a chave da rodada e os bits de entrada.

```
        def add_round_key(state, round_key):
            return [state[i] ^ round_key[i] for i in range(16)]
```

3.3 Decifração

Para a decifração, o algoritmo segue as mesmas etapas da cifragem, porém em ordem inversa e com a inversão de algumas operações. A decifração utiliza as chaves na ordem inversa. A operação **AddRoundKey** permanece inalterada, uma vez que a operação XOR é auto-inversível. A etapa ShiftRows é substituída por **InvShiftRows**, onde as linhas do estado são deslocadas de volta para suas posições originais. A etapa SubBytes é substituída por **InvSubBytes**, que utiliza uma tabela de substituição inversa. Por fim, a operação MixColumns utiliza a matrix inversa **matrizInversaMixColumns** correspondente.

3.4 Conclusão

O trabalho com o algoritmo AES (Advanced Encryption Standard) revela a complexidade de um dos métodos de criptografia mais amplamente utilizados no mundo. A estrutura do AES, com suas etapas de substituição, permutação e mistura, bem como a aplicação repetida de rodadas de transformação, demonstra um equilíbrio cuidadoso entre segurança e eficiência.

Referências

[1] Wikipédia. *Advanced Encryption Standard* - Wikipédia, a enciclopédia livre. Disponível em: https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard). Acesso em: 28/08/2024

[2] Wikipédia. *Cifra de bloco* - Wikipédia, a enciclopédia livre. Disponível em: https://pt.wikipedia.org/wiki/Cifra_de_bloco. Acesso em: 28/08/2024

[3] TREVISAN, D. F., SACCHI R. P. S., SANABRIA L. *Estudo do Padrão Avançado de Criptografia AES –Advanced Encryption Standard* - RITA, Volume 20, Número 1, 2013 https://seer.ufrgs.br/index.php/rita/article/view/rita_v20_n1_p13/23763. Acesso em: 29/08/2024

[4] EDUARDO, V., BONA L. C. E., ZOLA W. M. N. *Utilização de Criptografia em Modo CTR Aplicada ao Armazenamento de Arquivos em Nuvem* - Sociedade Brasileira de Computação https://sol.sbc.org.br/index.php/wcga/article/download/2374/2338/#:~:text=J%C3%A1%20o%20modo%20CTR%20consiste,XOR%20resulta%20no%20texto%20cifrado.. Acesso em: 29/08/2024

[5] MEDIUM *AES encryption with python step by step* https://femionewin.medium.com/aes-encryption-with-python-step-by-step-3e3ab0b0fd6c.. Acesso em: 29/08/2024