Implementando o Advanced Encryption Standard

Eduardo Freire dos Santos, 211010299
Ruan Petrus Alves Leite, 211010459

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

dudufreiresantos@gmail.com, 211010459@aluno.unb.br

1. Introdução

Esse projeto apresenta o Advanced Encryption Standard (AES) implementando tanto sua etapa de cifragem quanto de decifragem utilizando blocos e chaves de 128 bits. Ambas essas funcionalidades são implementadas tanto para blocos quanto para mensagens longas de texto, utilizando para essa última o modo de operação CTR.

2. Funcionamento do Algoritmo

O AES cifra blocos de 16 bytes, organizados em uma matriz (chamada de estado) da seguinte forma [Dworkin 2001b]:

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

O AES funciona por meio da execução sequencial de diversas etapas que transformam o estado, descritas nas subseções que seguem.

2.1. Key Expansion

Cada uma das etapas de transformação de estado necessitam de uma chave diferente. Como é fornecida apenas uma chave de 128 bits, há primeiro um processo de expansão dessa chave para R chaves, em que R é a quantidade de etapas (chamadas de "rounds") a serem executadas.

2.2. Add Round Key

Nessa etapa a chave do round atual é combinada com a matriz do estado coluna por coluna. Para isso é feito o XOR bitwise de cada byte do estado com cada byte da chave atual.

2.3. Substitute Bytes

Nessa etapa cada byte é substituído de acordo com uma tabela chamada de "Rijndael S-box" [Daemen 1999]. Essa tabela é escolhida de forma a propositalmente minimizar a correlação entre transformações lineares dos bits de input e output.

2.4. Shift Rows

Cada linha da matriz de estado é rotacionada ciclicamente uma certa quantidade de vezes à esquerda. A primeira é rotacionada 0 vezes, a segunda 1 vez, a terceira duas vezes e a última 3 vezes.

2.5. Mix Columns

Nesse passo cada coluna da matriz de estados é substítuida por sua multiplicação com uma matriz específica [Dworkin 2001b]. Essa multiplicação é feita de tal forma que os coeficientes da matriz são interpretados como polinômios no corpo finito de 2^8 elementos.

2.6. Funcionamento Geral

Primeiramente é feita a expansão de chave para que hajam chaves o suficientes para cada um dos R rounds. Em seguida, é feito o Add Round Key.

No próximo momento são executadas as etapas Substitute Bytes, Shift Rows, Mix Columns e Add Round Key sequencialmente $R - 1$ vezes.

Por fim, são executa-se Substitute Bytes, Shift Rows e Add Round Key. O estado final é o output do algoritmo.

Vale notar que a decifragem é feita de maneira muito similar a cifragem, apenas realizando os inversos das etapas descritas na ordem contrária.

2.7. O modo CTR

O algoritmo apresentado acima trabalha apenas com inputs de tamanho exatamente igual a 128 bits. O modo CTR [Dworkin 2001a] descreve uma maneira de estender-lo para inputs de tamanho arbitrário.

Para isso é escolhido um número de 128 bits inicial chamado de Counter com algumas restrições relacionadas a sua unicidade em termos da chave. A partir desse são gerados outros counters simplesmente incrementando o counter inicial. Outras formas de obter os counters mais complexas que incrementação são possíveis, porem menos populares.

Tendo os counters, separamos o input em blocos, cada um com tamanho 128 bits exceto potencialmente o último. Para obter a cifra desse bloco, ciframos o counter com a chave especificada e realizamos um XOR com o bloco de texto.

3. Implementação

Todas as etapas descritas acima podem ser implementadas de maneira trivial com exceção da Key Expansion e Mix Columns.

3.1. Key Expansion

A implementação dessa etapa foi guiada fortemente pelo pseudo-código descrito em [Daemen 1999] na página 16. Nela são feitas algumas transformações como rotação cíclica de chaves anteriores e substituição utilizando a S-Box descrita acima.

3.2. Mix Columns

Para essa etapa foi necessário implementar a multiplicação em $GL(2^8)$ da forma como é sugerido em [Dworkin 2001b]. A implementação resultante pode ser vista na Figura 1.

```
def xtime(a, n):
    if n == 0: return a
    a = xtime(a,n-1)

    msb = a >> 7
    result = (a << 1) % 256
    if msb == 1:
        result ^= 0x1B
    return result

def mul(a,b):
    result=0
    for i in range(8):
        if a & 1 == 1:
            result ^= xtime(b, i)
        a >>= 1
    return result
```

Figure 1. Implementação da multiplicação em $GL(256)$

4. Testes

Em [Dworkin 2001b] a partir da página 41 existem diversos exemplos do funcionamento correto do algoritmo AES. Comparando o resultado do algoritmo implementando no projeto com os resultados esperados descritos no artigo, verificamos que esse de fato está correto. Por exemplo, o artigo cita que o plaintext $00112233445566778899aabbccddeeff$ com chave $000102030405060708090a0b0c0d0e0f$ deve ser cifrado para $69c4e0d86a7b0430d8cdb78070b4c55a$, que de fato foi o caso.

Além disso, em [Dworkin 2001a] encontram-se exemplos da operação no modo CTR, que novamente concordam com os resultados obtidos no projeto.

Na última página seguem imagens do personagem Mário criptografadas com quantidades diferentes de rounds do AES.

References

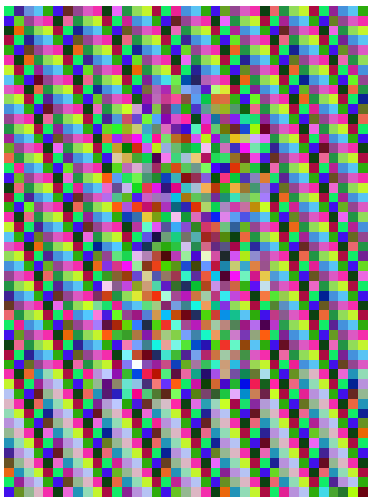
[Daemen 1999] Daemen, J. (1999). Aes proposal: Rijndael. https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1. [Online; accessed 30-October-2023].

[Dworkin 2001a] Dworkin, M. (2001a). Recommendation for block cipher modes of operation. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf. [Online; accessed 30-October-2023].

[Dworkin 2001b] Dworkin, M. (2001b). Specification for the advanced encryption standard (aes). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf. [Online; accessed 30-October-2023].

Figure 2. Resultado da encryptação na imagem do mario combined with Aliev-Panfilov.

(a) original



(b) counter=1

(c) counter=5

(d) counter=9

(e) counter=13