Relatório do Trabalho Final

Análise Comparativa entre os Algoritmos de Criptografia DES e AES

Discentes: Breno Cupertino, Gabriel Baptista, Yan Brandão

Professor: Dr. George Lima

Disciplina: Estruturas de Dados e Algoritmos II

Link do repositório com código fonte (GitHub)

Link de execução online do projeto (Replit)

Algoritmos Avaliados

Algoritmo DES (Data Encryption Standard)

O Des é um algoritmo de chave simétrica usado para criptografia de dados eletrônicos. A operação fundamental do Des envolve pegar um bloco de 64-bit (8 bytes) de texto simples como entrada e transforma-lo em um bloco de 64-bit de texto cifrado. O processo de criptografia envolve alguns passos, organizados em séries de rodadas juntamente com uma estrutura de rede Feistel. Segue abaixo a explicação de cada passo:

1. Permutação inicial (IP):

O texto simples de 64-bit sofrerá uma permutação inicial, onde as posições de seus bits são rearranjadas de acordo uma tabela predefinida.

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tabela IP utilizada na implementação

2. Geração de chave:

É aqui onde a chave de 56 bits é expandida e dividida em 16 subchaves, uma para cada rodada.

O processo começa com uma chave de 64 bits que é fornecida pelo usuário para iniciar o processo de criptografia. Na implementação, a chave é gerada de forma automática e possui tamanho inteiro de 16, após isso, ela passa por um processo em que é transformada em bloco de 64 bits.

Após a passagem da chave, 8 bits são jogados fora e os 56 bits restantes passam por uma permutação inicial e compressão. Durante essa etapa os bits são rearranjados de acordo com uma tabela predefinida e o resultado é uma permutação de 56 bits da chave original.

Permutation Table

	1	2	3	4	5	6	7	8
0	57	49	41	33	25	17	9	1
1	58	50	42	34	26	18	10	2
2	59	51	43	35	27	19	11	3
3	60	52	44	36	63	55	47	39
4	31	23	15	7	62	54	46	38
5	30	22	14	6	61	53	45	37
6	29	21	13	5	28	20	12	4

Tabela utilizada para permutação e compressão inicial da chave

Agora a chave de 56 bits é repartida em duas metades de 28 bits: a metade esquerda (E0) e metade direita (D0). Em cada rodada, ambas as metades são giradas ou deslocadas por um certo número de bits, utilizando uma escala de deslocamento que na implementação é a [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1], dando origem a uma nova combinação de bits.

Após a rotação as metades são concatenadas e comprimidas para criação de uma subchave de rodada de 48 bits para aquela rodada especifica, ou seja, para rodada 1 utilizaremos a chave K1. É responsabilidade da compressão mudar a chave de 56 bits para 48 bits, a tabela utilizada na compressão também é predefinida.

Key Compression Table

	1	2	3	4	5	6	7	8
1	14	17	11	24	01	05	03	28
2	15	06	21	10	23	19	12	04
3	26	08	16	07	27	20	13	02
4	41	52	31	37	47	55	30	40
5	51	45	33	48	44	49	39	56
6	34	53	46	42	50	36	29	32

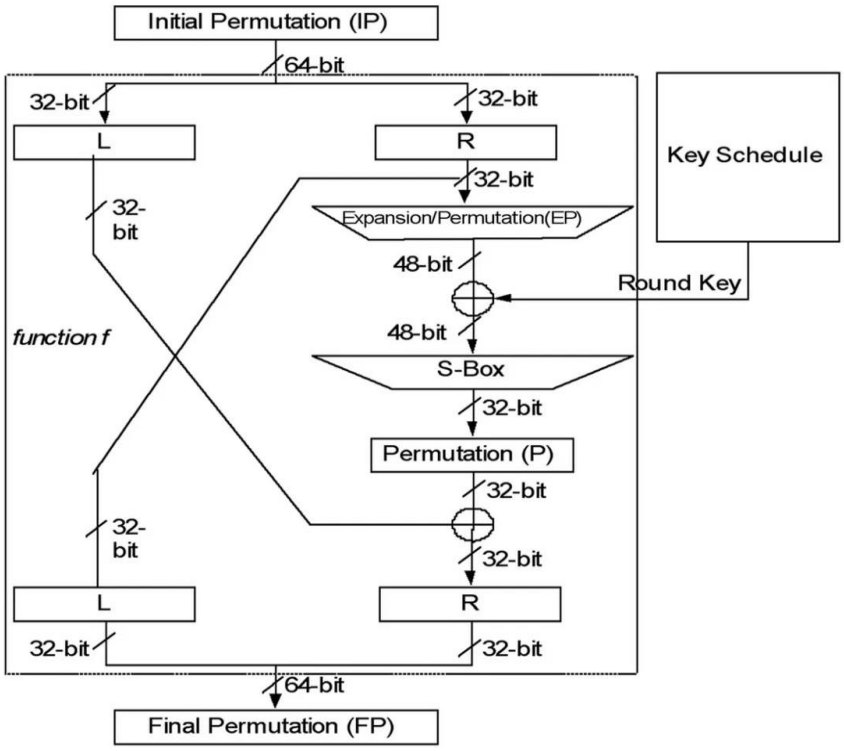
Tabela de compressão utilizada na implementação

Agora é necessário realizar o processo de expansão da chave, envolve criar 16 subchaves especificas para cada rodada, ou seja, para E1 e D1 utilizaremos E0 e D0, e continuaremos assim até E15 e D15.

Esse processo é repetido para cada uma das 16 rodadas, resultando em um conjunto de 16 subchaves de 48 bits (K1, K2, ..., K16).

3. Rodadas de criptografia

O bloco de dados é dividido em duas partes (L e R) e cada rodada envolve a transformação de uma metade com base na outra metade e na chave da rodada. Uma imagem ilustrativa ajudará a entender todo o processo feito em cada rodada.



Processo realizado em cada rodada

A parte direita (R) sofre uma operação de expansão, aumentando seu tamanho de 32 bits para 48 bits. A parte direita expandida é, então, sujeita a uma operação do tipo XOR (OR exclusivo) com a chave especifica daquela rodada (K_i) que foi derivada do processo de geração de chave.

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tabela utilizada para expansão

O resultado da operação XOR é dividido em 8 blocos de 6 bits. Cada bloco é alimentado em uma S-box (caixa de substituição), o trabalho dessas tabelas S-boxes pré-definidas é transformar esse bloco de 6 bits em uma saída de 4 bits. As saídas de todas as S-boxes são concatenadas para geração de uma nova saída de 32 bits.

A saída de 32 bits das S-boxes é sujeita a uma permutação fixa (permutação P-box). Acontecerá a reorganização dos bits de acordo com uma tabela de permutação predeterminada.

P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tabela utilizada para permutação

A saída da permutação sofre uma operação XOR com a parte esquerda original (L0). O resultado dessa operação se torna a nova metade direita (R1) e a metade direita original (R0) se torna a nova parte esquerda (L1). Para a segunda rodada,

repetiremos esses passos com sua chave especifica nas metades L1 e R1 e faremos isso até completarmos as 16 rodadas.

4. Permutação final (FP):

Após a 16º rodada, as metades esquerda e direita são trocadas uma última vez. O resultado final sofrerá uma permutação final (será o inverso da permutação inicial) para geração do bloco de texto crifrado de 64 bits.

Algoritmo AES (Advanced Encryption Standard)

O AES é uma cifra simétrica de bloco que pretende substituir o DES como o padrão para uma grande variedade de aplicações. A cifra recebe como entrada um bloco de texto sem formatação de tamanho 128 bits, ou 16 bytes. O comprimento da chave pode ser 16, 24 ou 32 bytes (128, 192 ou 256 bits). O algoritmo é denominado AES-128, AES-192 ou AES-256, dependendo do tamanho da chave.

A entrada para o algoritmo de encriptação e decriptação é um único bloco de 128 bits esse bloco é indicado como uma matriz quadrada de bytes 4 × 4, esse bloco é copiado para um array Estado, que é modificado a cada etapa de encriptação ou decriptação.

A cifra consiste em N rodadas, e o número delas depende do comprimento da chave: 10 rodadas para uma chave de 16 bytes, 12 para uma chave de 24 bytes e 14 para uma chave de 32 bytes.

As primeiras N – 1 rodadas consistem em quatro funções de transformação distintas: SubBytes, ShiftRows, MixColumns e AddRoundKey. A rodada final contém apenas três transformações, e há uma transformação inicial única (AddRoundKey) antes da primeira rodada, o que pode ser considerado Rodada 0.

Cada transformação usa uma ou mais matrizes de 4 × 4 como entrada e produz uma de 4 × 4 como saída.

Além disso, a função de expansão de chave gera N + 1 chaves de rodada, cada uma das quais é uma matriz distinta de 4 × 4. Cada chave de rodada serve como uma das entradas para a transformação AddRoundKey em cada rodada.

A seguir, serão explicadas cada fase de uma rodada:

SubBytes:

Nesta fase, o AES define uma matriz de 16 × 16 de valores de byte, chamada de S-box que contém uma permutação de todos os 256 valores possíveis de 8 bits.

Cada byte individual de Estado é mapeado para um novo byte da seguinte maneira: os 4 bits mais à esquerda do byte são usados como um valor de linha e os 4 bits mais à direita, como um valor de coluna.

Esses valores de linha e coluna servem como índices para a S-box a fim de selecionar um valor de saída de 8 bits.

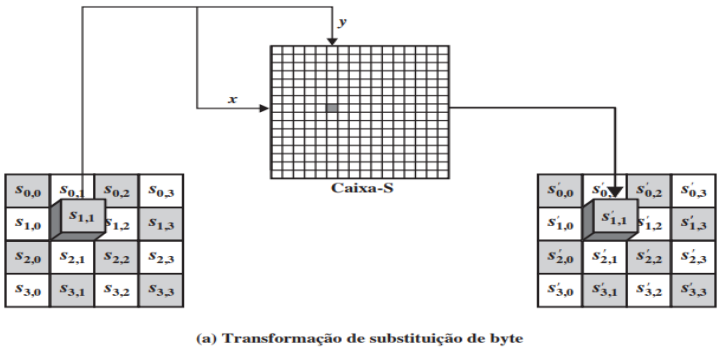


Imagem representando uma transformação de substituição de byte

A transformação invertida de substituição de byte, chamada InvSubBytes funciona de maneira análoga, só que para o processo de decriptografia e utiliza a S-box inversa.

ShiftRows:

Nesta fase, a primeira linha de Estado não é alterada. Para a segunda linha, é realizado um deslocamento circular à esquerda por 1 byte. Para a terceira linha, é feito um deslocamento circular à esquerda por 2 bytes. Para a quarta linha, ocorre um deslocamento circular à esquerda por 3 bytes.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Imagem representando uma transformação de permutação de colunas

A transformação inversa de deslocamento de linhas, chamada InvShiftRows, realiza o processo de permutação de colunas para a decriptografia. Os deslocamentos circulares são realizados na direção oposta para cada uma das três últimas linhas, com um deslocamento circular à direita por um byte para a segunda linha, e assim por diante.

MixColumns:

A transformação direta de embaralhamento de colunas, chamada MixColumns, opera sobre cada coluna individualmente. Cada byte de uma coluna é mapeado para um novo valor que é determinado em função de todos os quatro bytes nessa coluna. A transformação pode ser definida pela seguinte multiplicação de matriz sobre Estado:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Imagem representando uma transformação de mixagem de colunas

A matriz que está mais à esquerda é chamada de matriz MixColumns e é constante, isto é, já é definida pelo algoritmo AES.

A transformação inversa de embaralhamento de colunas, chamada InvMixColumns, que realiza o processo para a decriptografia, é definida pela seguinte multiplicação de matrizes:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Imagem representando uma transformação inversa de mixagem de colunas

A matriz mais à esquerda também já é definida pelo próprio algoritmo AES, em que a imagem acima é o inverso da imagem anterior.

AddRoundKey:

Na transformação direta de adição de chave da rodada, chamada AddRoundKey, os 128 bits de Estado passam por um XOR bit a bit com os 128 bits da chave da rodada.

A operação é vista como uma do tipo coluna por coluna entre os 4 bytes da coluna Estado e uma word da chave da rodada; ela também pode ser vista como uma operação em nível de byte. A seguir está um exemplo de AddRoundKey, em que a primeira matriz é o Estado, e a segunda, a chave da rodada:

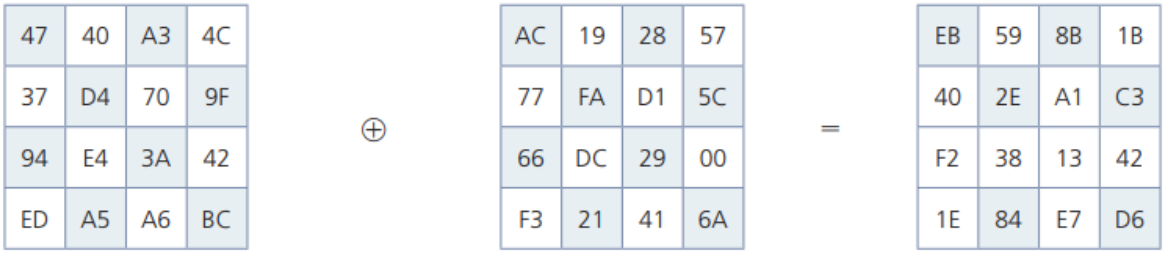


Imagem representando uma transformação de adição de chave de rodada

A transformação inversa de adição de chave da rodada é idêntica à direta de adição de chave da rodada, pois a operação XOR é o seu próprio inverso.

Ao final de todas as rodadas, o algoritmo produzirá o dado criptografado, no caso da criptografia, ou decriptografado, no caso da decriptografia.

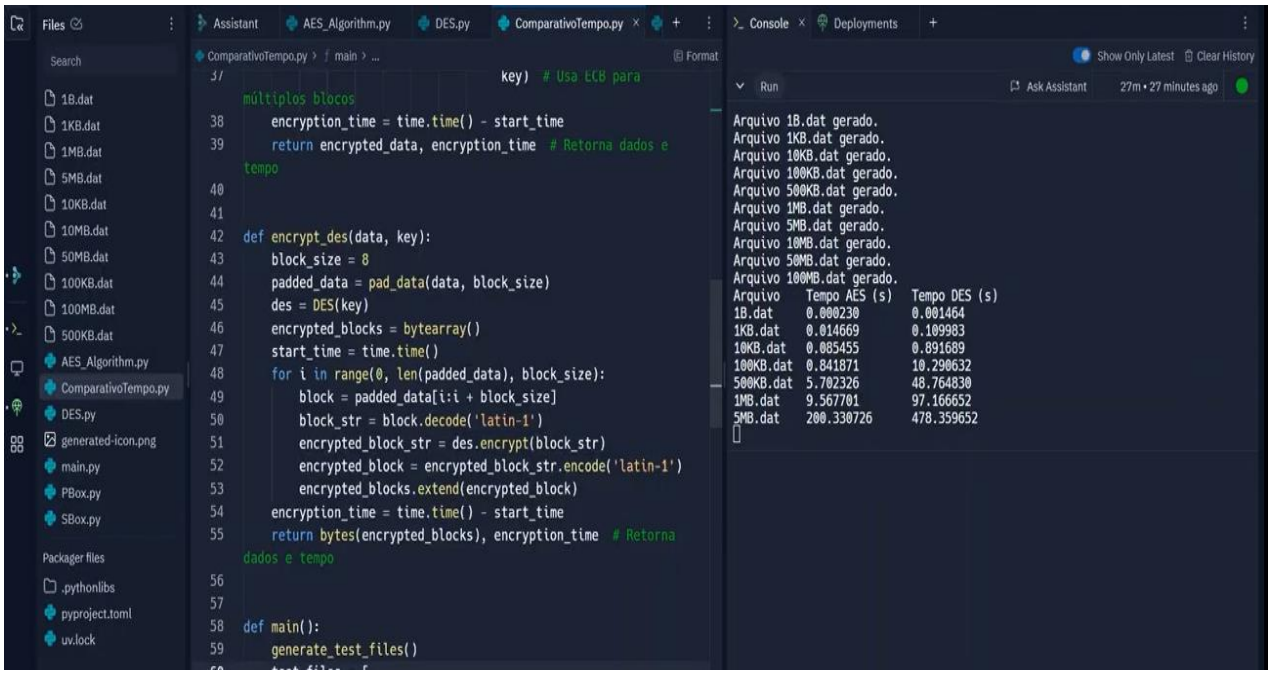
Método de Avaliação

O método de avaliação comparativa entre DES e AES escolhido foi por tempo (em segundos) que cada um dos nossos algoritmos levava para encriptar arquivos de tamanhos crescentes (1B, 1KB, 10KB, 100KB, 500KB, 1MB, 5MB, 10MB, 50MB, 100MB), por meio de um código que gera esses arquivos no momento de sua execução e chama os métodos desenvolvidos nesse trabalho para encriptar esses arquivos um de cada vez, enquanto conta quanto tempo cada algoritmo levou para chegar nesse resultado.

Dessa forma, num ambiente com CPU e RAM limitados, teremos um comparativo de como essas estruturas funcionam para exercer a mesma função, numa amostra

relativamente variada, condizente com o ambiente limitado que está trabalhado, visto que arquivos muito maiores demorariam muito mais tempo.

Resultados Alcançados



Resultados dos algoritmos AES e Des em diferentes tamanhos de arquivos

Conclusão

Em geral, assim como no nosso caso de teste, o AES é mais rápido e seguro do que o DES, e isso se deve a vários fatores:

Tempo de Encriptação

- **AES:**
  - **Desempenho:** O AES frequentemente conta com aceleração por hardware (como as instruções AES-NI) que permite encriptação em alta velocidade – chegando a processar gigabytes de dados por segundo em processadores mais atuais.
  - **Eficiência:** Mesmo em implementações puramente em software, o design do AES permite uma execução eficiente.
- **DES:**

- **Desempenho:** Embora o algoritmo em si seja relativamente simples, o DES não se beneficia do mesmo nível de otimizações modernas que o AES. Assim, a encriptação pode ser mais lenta, especialmente em grandes volumes de dados.
- **Observação:** Em arquivos pequenos, a diferença pode ser insignificante, mas em aplicações de alto desempenho, o DES decai rapidamente em eficiência.

Dificuldade de Implementação

- **AES:**
  - **Complexidade:** O algoritmo possui uma estrutura mais elaborada (com operações de substituição, permutação, mistura de colunas, etc.), mas sua ampla padronização e disponibilidade de bibliotecas e módulos otimizados tornam sua implementação direta para a maioria dos desenvolvedores.
  - **Suporte:** Devido à sua adoção mundial, existem inúmeras implementações seguras e auditadas, facilitando a integração em sistemas modernos.
- **DES:**
  - **Simplicidade:** O DES foi projetado em uma época anterior e, por isso, possui uma estrutura relativamente simples – o que pode facilitar a sua implementação em ambientes onde a simplicidade é priorizada.
  - **Limitações:** Entretanto, essa simplicidade vem acompanhada de limitações sérias em termos de segurança, tornando sua implementação insegura para a maioria das aplicações atuais.

Outros Fatores de Segurança

- **AES:**
  - **Segurança:** Opera com blocos de 128 bits e suporta chaves de 128, 192 ou 256 bits, oferecendo uma proteção robusta contra ataques de força bruta e outras técnicas criptográficas.
  - **Resistência a Ataques:** Até o momento, o AES não possui vulnerabilidades práticas que comprometam sua segurança, sendo considerado o padrão para criptografia simétrica.

- **DES:**
    - **Segurança:** Utiliza blocos de 64 bits e uma chave efetiva de 56 bits, o que é considerado insuficiente nos padrões atuais. Técnicas modernas de quebra por força bruta tornam o DES obsoleto para proteger informações sensíveis.

Conclusão

- **Desempenho:** Em sistemas modernos, o AES geralmente encripta arquivos mais rapidamente do que o DES, especialmente quando se utiliza aceleração por hardware.
- **Implementação:** Apesar de o DES ser mais simples de implementar devido ao seu design antigo, o AES é amplamente suportado e possui implementações robustas que compensam sua complexidade.
- **Segurança:** O AES oferece uma segurança muito superior, com chaves maiores e um design resistente a ataques, enquanto o DES é considerado inseguro para a maioria dos usos atuais.

Portanto, para aplicações modernas que exigem rapidez e alto nível de segurança, o AES é a escolha preferencial em comparação ao DES.

Referências

STARLLINGS, Willian. Criptografia e Segurança de Redes Princícios e Práticas. 6. ed.URL: https://www.kufunda.net/publicdocs/Criptografia%20e%20Seguran%C3%A7a%20de%20Redes%20-%206%C2%AA%20Ed.%202014.pdf

TOLDINAS, J.; ŠTUIKYS, V.; ZIBERKAS, G.; NAUNIKAS, D. Power Awareness Experiment for Crypto Service-Based Algorithms. **Elektronika ir Elektrotechnika**, *[S. l.]*, v. 101, n. 5, p. 57-62, 2010. Disponível em: https://eejournal.ktu.lt/index.php/elt/article/view/9426. Acesso em: 15 feb. 2025.