Trabalho 1 - Criptografia

Guilherme Eduardo Gonçalves da Silva
Curitiba, Brasil
gegs23@inf.ufpr.br

Heloísa de Oliveira Pinho
Curitiba, Brasil
hop23@inf.ufpr.br

I. Introdução

Este relatório aborda a capacidade de algoritmos de aprendizado de máquina em classificar cogumelos como comestíveis ou venenosos com base em características físicas observáveis. A correta identificação desses organismos é essencial, pois erros nesse processo podem trazer riscos à saúde humana.

Para o estudo, utilizou-se o conjunto de dados *Mushroom*, que reúne propriedades como formato e cor do chapéu, odor, características das brânquias, tipo de anel e textura do estipe. A variável alvo, *class*, indica se o cogumelo é *comestível* (e) ou *venenoso* (p), caracterizando um problema de classificação binária.

Foram avaliados quatro algoritmos supervisionados: Árvore de Decisão, k-Nearest Neighbors (k-NN), Naive Bayes Categórico e Perceptron Multicamadas (MLP). A escolha desses modelos se deve tanto à sua relevância na literatura quanto ao fato de terem sido estudados na disciplina, permitindo aplicar na prática os conceitos vistos em aula.

As próximas seções apresentam o processo metodológico adotado, os modelos utilizados e a análise dos resultados obtidos ao longo do experimento.

II. Nossa proposta

Diante dos estudos de algoritmos clássicos de criptografia simétrica, na qual a mesma chave é utilizada tanto para a cifragem quanto para a decifragem, o nosso algoritmo, denominado *Pizão*, foi desenvolvido a partir de inspirações nas técnicas da *Cifra de Playfair* e da *Cifra de César*.

Playfair é um algoritmo de criptografia do tipo substituição, no qual pares de caracteres do texto simples são substituídos por pares alternativos de letras. O método utiliza uma matriz 5×5, construída a partir de uma palavra-chave, que acomoda 25 letras do alfabeto (normalmente unificando as letras “I” e “J”). Essa matriz define as regras de substituição, de modo que o texto simples deve ser dividido em pares de letras antes da aplicação do processo de cifragem [1].

A cifra de César funciona com um simples princípio de substituição. Tal que, cada letra no texto simples é substituída por uma letra um número fixo de posições no alfabeto. Por exemplo, a aplicação de um deslocamento de 4 para a palavra ”COMPUTER”produz o texto cifrado ”GSQTYXIV”, onde cada letra é deslocada 4 posições para a frente no alfabeto. Essa transformação obscurece a mensagem original de leitores não autorizados. Para descriptografar ”GSQTYXIV”, o destinatário simplesmente desloca cada letra para trás por 4 posições [2]

Com base nesses fundamentos, o algoritmo *Pizão* propõe uma abordagem alternativa para os processos de cifragem e decifragem, combinando esses princípios.

A. Cifra - Pizão

Para a realização da cifra, inicialmente é necessária a definição de uma chave, a partir da qual será construída a matriz de *PlayFair*.

Algorithm 1 Pseudocódigo do Processo de Cifragem

```
Entrada: Texto claro T, chave K
Saída: Mensagem Cifrada C

M ← PlayfairMatriz(K)
letra ← matriz[1,5]
desloc ← SomaDigitos(ASCII(letra))
M ← Transp(M, desloc)          // transposição triangular
T ← cifraPlayfair(M, T)
C ← cifraCesar(T, desloc)
Retornar C
```

Com a matriz formada, seleciona-se a última letra da primeira linha para obter o seu valor na tabela ASCII (entre 65, correspondente à letra A, e 90, correspondente à letra Z). Essa letra é então substituída pelo primeiro elemento da mesma linha.

Em seguida, realiza-se a soma dos dois valores ASCII — por exemplo, para a letra I (73), a operação de soma e posterior separação dos dígitos resulta em $1 + 0 = 1$ e $7 + 3 = 10$. Esse valor servirá como índice para uma nova substituição: o elemento atualmente na última posição da primeira linha é trocado pelo símbolo que originalmente ocupava a posição correspondente ao valor obtido. Dessa forma, estabelece-se uma relação triangular entre os valores envolvidos. Podemos observar esse processo conforme 1.

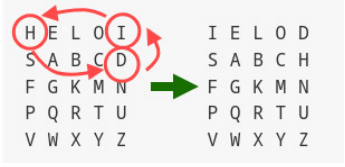


Figura 1. Matriz PlayFair gerada a partir da chave utilizada.

Após a finalização da matriz de *PlayFair*, cifraremos o texto claro utilizando o próprio método de substituição dessa

técnica. E por último, aplica-se uma cifra de César com o número de rotação baseado na mesma letra, a qual foi efetuada a soma conforme os números representados pela tabela ASCII. O Algoritmo 1 apresenta todo o processo de cifra

B. Decifra - Pizão

Para o processo de decifra, realiza-se a sequência inversa da cifra, iniciando pela inserção da chave, logo em seguida, a construção da matriz de *PlayFair* para obtenção do valor do carácter da tabela ASCII. A partir desse valor, é possível iniciar todo o processo. O algoritmo 2 ilustra o procedimento.

Algorithm 2 Pseudocódigo do Processo de Decifragem

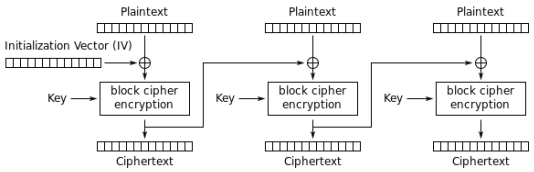
```
Entrada: Texto Cifrado T, Chave K
Saída: Texto Decifrado D

M ← PlayfairMatriz(K)
letra ← matriz[1,5]
desloc ← SomaDigitos(ASCII(letra))
C ← decifraCesar(T, desloc)
M ← Transp(M, desloc)          // transposição triangular
D ← decifraPlayfair(M, C)
Retornar D
```

III. AES

Após o algoritmo DES ser considerado inseguro em 2001 pelo NIST (*National Institute of Standards and Technology*), o *Advanced Encryption Standard*(AES) ou também conhecido como *Rijndael*, e passou a ser adotado por muitas organizações em todo o mundo [3].

O AES opera em blocos de 128 bits de dados que são de comprimento do bloco de entrada, do bloco de saída e do Estado. O algoritmo pode criptografar e descriptografar blocos usando chaves secretas. O tamanho da chave pode ser 128 bits, 192 bits ou 256 bits. O tamanho real da chave depende do nível de segurança desejado. As diferentes versões são mais frequentemente denotadas como AES-128, AES-192 ou AES-256.

Neste relatório, optou-se pelo uso do modo AES-CBC (*Cipher Block Chaining*) usa *feedback* para alimentar o resultado da criptografia de volta à criptografia do próximo bloco. No modo CBC, cada bloco de texto simples é combinado, por meio da operação XOR, com o bloco de texto cifrado anterior antes de ser processado pelo algoritmo. Desta forma, cada bloco de texto cifrado depende de todos os blocos de texto simples processados até aquele ponto. Além disso, para tornar cada mensagem única, um IV deve ser usado no primeiro bloco. O IV não precisa ser mantido em segredo. O IV deve ser um número aleatório (ou um número de série), para garantir que cada mensagem seja criptografada exclusivamente [4]. A figura abaixo detalhe o processo de cifra.



Cipher Block Chaining (CBC) mode encryption

IV. Experimento

Os testes foram baseados na cifra e decifra de 3 tamanhos de arquivos, sendo eles de 1,4 MB, 100 KB e 10 KB, sendo todos disponibilizados em formato *.txt* codificados em UTF-8. Para efetuar a comparação justa dos algoritmos mencionados nesse trabalho, efetuamos a implementação de ambos em *python3*. Para a implementação do AES foi utilizada a biblioteca *PyCryptodome*, que fornece funções de criptografia eficientes.

Observando os resultados obtidos nas tabelas I e II, podemos observar a superioridade do AES em relação ao *Pizao*, como por exemplo, conseguindo ser 338 vezes mais rápido (1.150234 / 0.003397) a cifra no melhor caso para o teste com o livro *Moby Dick; Or, The Whale*. Além disso , o AES consegui atingir 149 microssegundos com arquivo de 10KB.

Tabela I
Comparação de tempos de cifra entre *Pizao* e AES

Algoritmo	**Livro**	**Tamanho**	**Tempo (s)**
Pizao	*Moby Dick; Or, The Whale*	1,4 MB	1.150234
AES	*Moby Dick; Or, The Whale*	1,4 MB	0.003397
Pizao	*Romeo and Juliet*	100 kB	0.148884
AES	*Romeo and Juliet*	100 kB	0.000533
Pizao	*Moby Multiple Language*	10 kB	0.021083
AES	*Moby Multiple Language*	10 kB	0.000198

Tabela II
Comparação de tempos de decifra entre *Pizao* e AES

Algoritmo	**Livro**	**Tamanho**	**Tempo (s)**
Pizao	*Moby Dick; Or, The Whale*	1,4 MB	1.128570
AES	*Moby Dick; Or, The Whale*	1,4 MB	0.003181
Pizao	*Romeo and Juliet*	100 kB	0.145015
AES	*Romeo and Juliet*	100 kB	0.000481
Pizao	*Moby Multiple Language*	10 kB	0.020527
AES	*Moby Multiple Language*	10 kB	0.000130

Abaixo, os gráficos em escala logarítmica facilitam a compreensão da diferença de magnitudade dos valores obtidos durante a execução dos algoritmos.

V. Conclusão

Os resultados obtidos evidenciam que, apesar de o algoritmo Pizão ser funcional e cumprir o propósito de ilustrar técnicas de cifragem e decifragem, o AES demonstrou desempenho muito superior, chegando a ser centenas de vezes mais eficiente em arquivos maiores. Isso explica sua ampla utilização em cenários reais, enquanto o Pizão se destaca principalmente como proposta didática.

Dessa forma, entende-se que o Pizão e válido como exercício acadêmico, permitindo explorar conceitos de

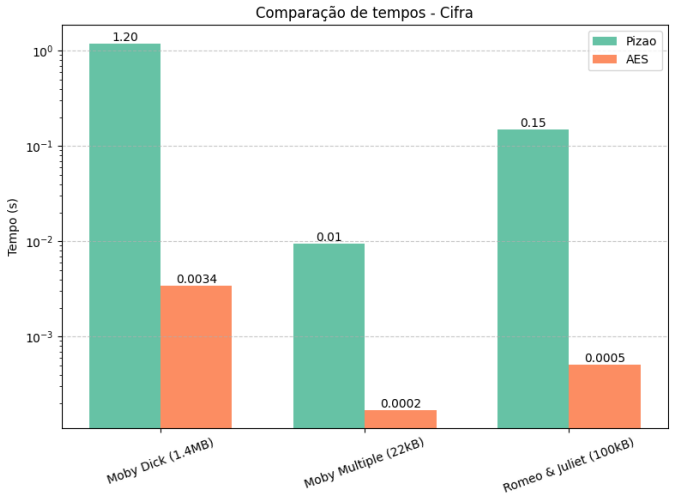


Figura 2. Gráfico de comparação de tempos de cifra entre *Pizao* e AES em escala logarítmica

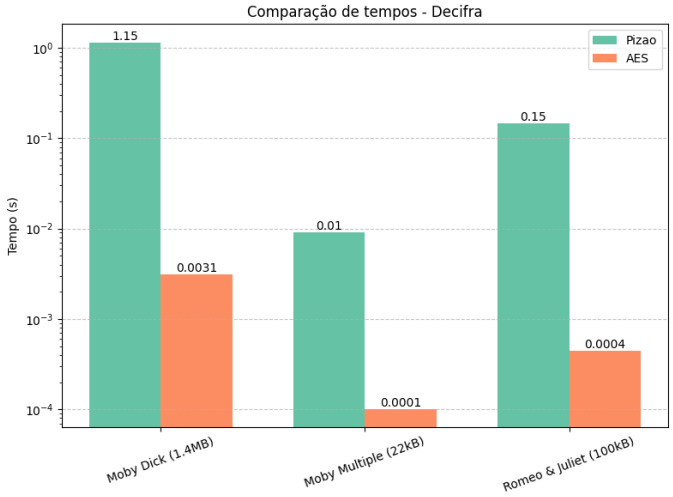


Figura 3. Gráfico de comparação de tempos de decifra entre *Pizao* e AES em escala logarítmica

substituição, transposição e sua influência na eficiência de um algoritmo. Além disso, a comparação realizada reforça a importância de avaliar não apenas a corretude, mas também o desempenho ao se considerar algoritmos de criptografia para aplicações práticas.

Referências

[1] J. C. C. Ferrer, F. E. D. Guzman, K. L. E. Gardon, R. J. R. Rosales, D. Dell Michael Badua, and D. R. Marcelo, "Extended 10 x 10 playfair cipher," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1–4.

[2] F. He and C. W. Chuah, "On the security of caesar cipher, linear cipher and enhanced linear cipher in hiragana messages," in *2025 IEEE 34th Wireless and Optical Communications Conference (WOCC)*, 2025, pp. 194–198.

[3] M. Vaidehi and B. J. Rabi, "Design and analysis of aes-cbc mode for high security applications," in *Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014*, 2014, pp. 499–502.

[4] H. Lee, K. Lee, and Y. Shin, "Implementation and performance analysis of aes-128 cbc algorithm in wsns," in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, vol. 1, 2010, pp. 243–248.