Análise comparativa de algoritmos de criptografia

Fernando Abreu de Sousa[1]

[1]Instituto de Engenharia e Geociências - Universidade Federal do Oeste do Pará (UFOPA)

Resumo. *Este meta-artigo descreve uma análise experimental comparativa entre diferentes algoritmos de criptografia, com foco na avaliação de desempenho, uso de memória, entropia e tamanho da saída. O estudo utiliza os algoritmos AES, 3DES e RSA aplicados a diversos tipos de dados. A metodologia segue um modelo com testes realizados em ambiente no Google Colab. Os resultados indicam que o AES é o algoritmo mais eficiente, equilibrando desempenho e segurança, sendo ideal para grandes volumes de dados.*

1. Introdução

A segurança da informação é um dos pilares fundamentais na era digital, especialmente com o crescimento exponencial do uso de dados sensíveis em diversas aplicações. Nesse contexto, os algoritmos de criptografia desempenham um papel crucial na proteção desses dados contra acessos não autorizados e ataques maliciosos. Diante disso, torna-se fundamental realizar uma análise prática e aprofundada da segurança oferecida por diferentes algoritmos criptográficos.

Este projeto tem como objetivo realizar uma análise experimental detalhada de alguns algoritmos de criptografia com base em critérios como desempenho, tamanho dos dados criptografados e memória utilizada. Para isso, será desenvolvido um estudo de caso no qual algoritmos selecionados serão submetidos a uma série de testes cuidadosamente planejados, visando identificar seus e pontos fortes e fracos.

Assim, o presente trabalho busca contribuir para uma visão mais clara e realista sobre a eficácia dos algoritmos criptográficos, destacando suas vantagens e limitações em diferentes contextos de aplicação. Espera-se, com isso, fornecer subsídios importantes para a escolha adequada de técnicas criptográficas em projetos que demandem alto nível de segurança.

O trabalho está organizados da seguinte forma, na seção 2 é mostrado uma visão geral sobre Criptografia Simétrica e Assimétrica. Na seção 3, é feita uma análise dos trabalhos relacionados, em seguida, a metodologia usado no trabalho. Os resultados são mostrado na seção 5, e por fim, as conclusões.

2. Criptografia

Antes de explorar os algoritmos, é importante entender alguns termos essenciais. De acordo com [Stallings 2014] mensagem original é chamada de texto claro, enquanto sua versão codificada é o texto cifrado. O processo de transformar texto claro em texto cifrado chama-se cifração ou encriptação, e o processo inverso é a decifração ou decriptação.

Os métodos usados para cifrar mensagens formam o campo da criptografia, e os esquemas em si são chamados de sistemas criptográficos ou cifras.

[Stallings 2014] também define duas formas de Criptografia, a simétrica e assimétrica, veremos o conceito de cada uma a seguir.

2.1. Modelo de Cifra Simétrica

Nos sistemas de cifra simétrica, a mesma chave é usada tanto para cifrar quanto para decifrar os dados. Um esquema típico de encriptação simétrica envolve cinco componentes:

Nos sistemas de cifra **simétrica**, a mesma chave é usada tanto para cifrar quanto para decifrar os dados. Um esquema típico de encriptação simétrica envolve cinco componentes:

1. **Texto claro**: a mensagem original legível.
2. **Algoritmo de encriptação**: aplica substituições e transformações ao texto claro.
3. **Chave secreta**: um valor independente do texto claro que determina o resultado da cifração.
4. **Texto cifrado**: a saída cifrada, que parece aleatória e não pode ser compreendida sem a chave.
5. **Algoritmo de decriptação**: opera de forma inversa ao de encriptação para restaurar o texto original.

Para garantir a segurança de um sistema simétrico, dois requisitos devem ser atendidos:

- O algoritmo de encriptação deve ser suficientemente forte para resistir a ataques, mesmo que o atacante conheça o algoritmo e possua acesso a exemplos de textos claros e cifrados.
- A chave secreta deve ser compartilhada de forma segura entre emissor e receptor, e mantida em sigilo absoluto.

2.2. Modelo de Cifra Assimétrica

Diferente da cifra simétrica, a **cifra assimétrica** (ou criptografia de chave pública) utiliza um par de chaves diferentes: uma **chave pública** e uma **chave privada**. Essa abordagem resolve o problema da distribuição segura da chave, comum nos sistemas simétricos.

Um esquema de cifra assimétrica envolve os seguintes elementos:

1. **Texto claro**: a mensagem original que se deseja proteger.
2. **Algoritmo de encriptação**: um algoritmo que utiliza a **chave pública** do destinatário para cifrar a mensagem.
3. **Chave pública**: pode ser livremente distribuída. Ela é usada para cifrar mensagens destinadas ao proprietário da chave correspondente.
4. **Texto cifrado**: o resultado da encriptação, que só pode ser decifrado com a chave privada.
5. **Algoritmo de decriptação**: usa a **chave privada** correspondente para restaurar o texto claro original.
6. **Chave privada**: deve ser mantida em segredo pelo destinatário. É a única capaz de decifrar as mensagens cifradas com a chave pública correspondente.

Funcionamento básico:

- Quando alguém deseja enviar uma mensagem segura, utiliza a **chave pública do destinatário** para cifrá-la.
- Apenas o destinatário, que possui a **chave privada**, poderá decifrá-la.
- Como as chaves são matematicamente ligadas, o que é cifrado com uma só pode ser decifrado com a outra.

3. Trabalhos Relacionados

Nesta seção, apresenta-se uma visão abrangente dos principais trabalhos prévios que tinham como tema a avalição de algoritmos de criptografia, tanto simétrica quanto assimétrica.

O estudo de [de Oliveira and de Freitas Júnior 2004] comparou o desempenho dos algoritmos de criptografia DES e IDEA, avaliando especialmente a velocidade de execução dos dois métodos implementados em linguagem C-ANSI em um ambiente Linux. Os autores analisaram as diferenças estruturais dos algoritmos e destacaram que o IDEA se mostrou mais eficiente por operar com blocos de 16 bits e evitar permutações excessivas, ao contrário do DES. Como resultado, o IDEA apresentou um desempenho aproximadamente 97,85% superior ao do DES em tempo de execução. Essa eficiência técnica, aliada a uma chave mais longa (128 bits contra 56 bits do DES), levou os autores a destacarem o IDEA como mais vantajoso, influenciando na escolha do algoritmo em aplicações que demandam maior velocidade e segurança.

O estudo de [Andrade and dos Santos Silva 2012] analisou a relação entre segurança e desempenho no algoritmo de criptografia RSA, observando o impacto do tamanho das chaves (1024, 2048 e 4096 bits) no tempo de cifragem e decifragem. Os autores implementaram o algoritmo em C utilizando a biblioteca GMP, simulando 1000 cifragens e 1000 decifragens para cada tamanho de chave. Como resultado, observaram que o aumento do tamanho da chave implica em crescimento exponencial no tempo de processamento, o que torna o RSA mais seguro, porém mais lento. Foi evidenciado que o módulo de 1024 bits, embora menos seguro, oferece desempenho significativamente superior, sendo mais indicado quando o tempo de resposta é crítico.

O estudo de [Oliveira 2012] avaliou os principais algoritmos de criptografia simétrica e assimétrica, descrevendo suas características, vantagens, desvantagens e aplicações no contexto da segurança da informação. O autor categorizou os algoritmos em dois grupos: os de chave privada (simétrica) e os de chave pública (assimétrica), abordando também certificados digitais, assinaturas digitais e funções de hashing. Para cada categoria, o autor listou os algoritmos mais relevantes (como AES, DES, RSA e Paillier), destacando seu funcionamento, segurança e desempenho. Como resultado, o trabalho apresenta uma visão abrangente e técnica das estratégias criptográficas disponíveis atualmente, enfatizando que a escolha do algoritmo deve considerar o equilíbrio entre segurança e eficiência de processamento, além da aplicação prática desejada. Tal abordagem foi fundamental para a compreensão e aplicação dos conceitos de segurança da informação utilizados no presente trabalho.

O estudo de [Lunkes and de Oliveira 2025] analisou a complexidade computacional dos algoritmos de Criptografia Parcialmente Homomórfica (PHE), avaliando o desempenho de cifragem, decifragem, homomorfismo e resistência a ataques. Os autores focaram em algoritmos que suportam exclusivamente operações aditivas ou multiplicativas sobre inteiros, como RSA, ElGamal, Paillier e Benaloh, entre outros. Cada algoritmo foi descrito detalhadamente, incluindo sua geração de chaves, funcionamento matemático e complexidade de execução. Como resultado, verificou-se que os esquemas apresentam diferentes níveis de desempenho e segurança, sendo que os baseados em adição (como Paillier e Benaloh) se destacaram em aplicações como votação eletrônica e armazenamento em nuvem. O estudo também destacou que a escolha do algoritmo ad-

equado depende da operação homomórfica desejada e da complexidade computacional aceitável. Este achado influenciou na escolha dos algoritmos utilizados no presente trabalho, especialmente em contextos que exigem manipulação de dados criptografados sem necessidade de decifragem.

O estudo de [Voitechen 2015] avaliou o desempenho dos algoritmos de criptografia AES, Blowfish, RC2, RC5, RC4, DES, 3DES (simétricos) e RSA (assimétrico) na proteção de imagens captadas por radares de trânsito, verificando quais algoritmos preservariam a integridade da imagem e apresentariam melhor desempenho em tempo de execução. A autora utilizou pacotes com diferentes quantidades de imagens e testou diversas chaves (128, 192 e 256 bits para algoritmos simétricos; 2048, 4096 e 16384 bits para RSA), comparando os resultados por algoritmo, tamanho de chave e volume de dados. Como resultado, verificou-se que todos os algoritmos simétricos mantiveram a integridade das imagens de forma satisfatória, sendo que os algoritmos RC4 e RC5 apresentaram melhor desempenho em tempo de processamento. Já o algoritmo RSA mostrou-se inviável para grandes volumes de dados, sendo recomendado apenas para pequenas mensagens. Faz-se importante pontuar que a combinação dos algoritmos RSA e RC5 demonstrou ser a mais eficiente para criptografar grandes volumes de bits, como imagens, conciliando segurança e desempenho.

4. Metodologia

Este trabalho adota uma abordagem experimental para avaliar a segurança e o desempenho de diferentes algoritmos criptográficos. A metodologia se baseia no processo chamado *Data Science Trajectories* (DST), proposto por [Martínez-Plumed et al. 2021]. O DST é um conjunto de trajetórias metodológicas que auxiliam em atividades de *Machine Learning*, ajudando a extrair informações relevantes de dados brutos. Nesse trabalho, o DST foi usado para selecionar uma abordagem metodológica que melhor se adequasse ao tema proposto, já que não se trata de um trabalho de *Machine Learning*. Nesse sentido, a metodologia proposta seguiu a seguinte estrutura:

1. **Compreensão do Domínio do Problema**: Inicialmente, foi realizada uma análise do papel da criptografia na segurança da informação. Foram investigadas as diferenças entre criptografia simétrica e assimétrica, bem como os principais desafios no uso prático dessas técnicas, especialmente no que diz respeito ao desempenho computacional. Essa compreensão fundamentou a definição dos critérios de avaliação utilizados no experimento.
2. **Exploração de Objetivos**: Com base no domínio identificado, o objetivo estabelecido foi avaliar e comparar algoritmos criptográficos (AES, RSA, DES) quanto à eficiência e comportamento frente a diferentes tipos de dados. Para isso, definiu-se um conjunto de métricas quantitativas — como tempo de execução, uso de memória, *throughput* e entropia da saída — a serem extraídas de forma sistemática em um ambiente de teste controlado. A análise foi pensada tanto do ponto de vista computacional quanto de segurança da informa
3. **Revisão da Literatura**: Foi realizada uma busca por materiais técnicos e artigos científicos que abordam algoritmos criptográficos clássicos e modernos, suas aplicações práticas e formas de mensuração de desempenho. A literatura consultada ajudou a embasar a escolha dos algoritmos, dos parâmetros (como modos

de operação e tamanho de chaves) e das métricas de avaliação, além de validar a importância da entropia como medida de aleatoriedade e segurança.

4. **Seleção dos Algoritmos**: Nesta etapa foram selecionados três algoritmos para a análise, sendo eles:
    - AES (*Advanced Encryption Standard*), representando a criptografia simétrica moderna;
    - RSA (*Rivest-Shamir-Adleman*), representando a criptografia assimétrica;
    - 3DES (*Triple Data Encryption Standard*), como exemplo de algoritmo simétrico clássico.
5. **Modelagem**: Para testar os algoritmos em diferentes cenários, foram escolhidos sete tipos de entrada: JSON, CSV, PDF, daddos já criptografados, textos curtos e longos e imagens.
6. **Avaliação**: Nesta fase, foi criado um ambiente de testes automatizado em *Python*. Para cada algoritmo e tipo de entrada, seguindo a seguinte estrutura em um *Notebook* no *Google Collab*:
    - Executa a criptografia dos dados com a biblioteca *criptografy*[1] do *Python*;
    - Mede o tempo e uso de memória dos algoritmos com o módulo *tracemalloc*[2] do *Python*;
    - Calcula o *throughput* em KB/s;
    - Mede a entropia;
    - Armazena os dados em uma estrutura para análise posterior bem como são gerados gráficos para tal.

5. Resultados

Partindo dos materiais e métodos estabelecidos e em consonância com os objetivos previamente definidos, foram avaliados três algoritmos criptográficos, AES, RSA e DES, aplicados a sete tipos distintos de entrada: JSON, CSV, arquivos PDF, imagem, textos curtos e longos e dados previamente cifrados. Cada combinação foi submetida a um processo de criptografia no qual foram coletadas métricas de desempenho, incluindo tempo de execução, uso de memória, *throughput*, entropia da saída e tamanho do resultado.

Antes de prosseguirmos para a análise dos resultados, faz-se necessário compreender as métricas utilizadas para avaliar o desempenho e a segurança dos algoritmos criptográficos aplicados. As principais métricas consideradas foram o **tempo de execução**, que indica a duração necessária para completar o processo de criptografia; o **uso de memória**, medido durante a execução para avaliar o impacto computacional de cada algoritmo, o **tamanho da saída**, que pode influenciar diretamente na eficiência de armazenamento e transmissão dos dados. Também foram avaliadas duas métricas que são amplamente utilizadas em análise de desempenho de algoritmos computacionais, são elas o ***throughput*** e a **entropia**. *Throughput*, ou vazão, é uma medida que indica a quantidade de dados que um algoritmo consegue processar por unidade de tempo, ou seja, representa a velocidade com que um algoritmo criptográfico consegue cifrar uma determinada quantidade de dados. Essa métrica é geralmente expressa em kilobytes por segundo (KB/s) e é obtida dividindo o tamanho da entrada pelo tempo total gasto na execução do algoritmo. Um *throughput* alto indica que o algoritmo é eficiente em termos de desempenho, sendo

[1] https://pypi.org/project/cryptography/

[2] https://docs.python.org/3/library/tracemalloc.html

capaz de processar rapidamente grandes volumes de dados. Entropia, por sua vez, é uma medida da aleatoriedade ou imprevisibilidade de um conjunto de dados. Na criptografia, a entropia da saída cifrada é usada como um indicativo de segurança: quanto mais aleatória a saída, mais difícil será para um atacante encontrar padrões que permitam a quebra da cifra. A entropia é calculada com base na distribuição de frequência dos símbolos presentes nos dados e, no caso de bytes, seu valor varia de 0 a 8 bits por byte. Uma saída com entropia próxima de 8 bits por byte é considerada altamente aleatória, o que é desejável em sistemas criptográficos seguros.

A seguir, apresentam-se os resultados obtidos para cada métrica utilizada, buscando analisá-los à luz da literatura relacionada.Os resultados foram organizados por meio de gráficos, permitindo observar o comportamento relativo dos algoritmos em diferentes contextos.

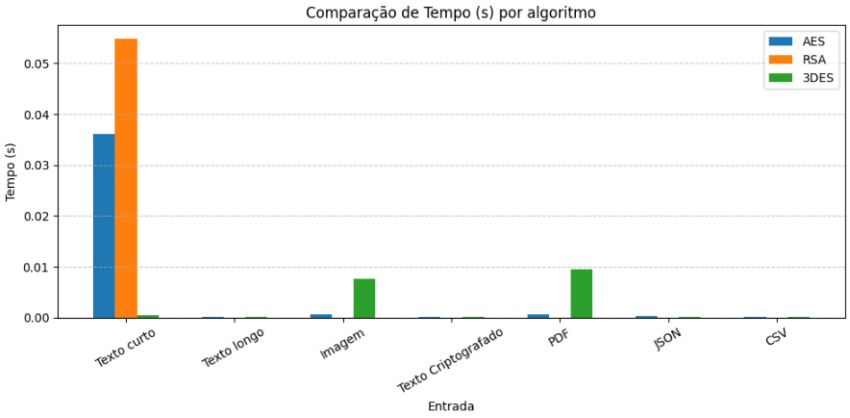

Figure 1. Comparação de Tempo por Algoritmo

A Figura 1 mostra os resultados[3] para o tempo de execução, e confirma o comportamento esperado de cada algoritmo. O RSA teve um tempo significativamente maior que os demais na entrada ”Texto curto”, com 0,0547 segundos, enquanto o 3DES executou em 0,00042 s e o AES em 0,0361 s. Para todas as outras entradas, o RSA não foi testado devido à sua limitação quanto ao tamanho de entrada, reforçando seu uso limitado a pequenos blocos de dados. O AES e o 3DES apresentaram tempos muito baixos em todas as entradas restantes. O AES, por exemplo, cifrou um arquivo PDF de quase 194 KB em apenas 0,00061 segundos, enquanto o 3DES levou 0,0095 s para o mesmo arquivo, evidenciando a superioridade do AES em velocidade.

[3]Se quiser ver os resultados de forma mais abrangente a Apêndice A mostra uma tabela com todos os valores para cada entrada e algoritmo

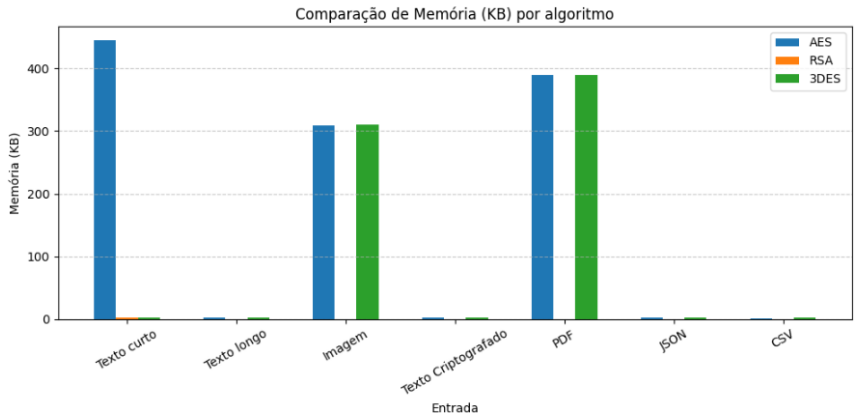

Figure 2. Comparação de Memória por Algoritmo

A utilização de memória foi bastante variável conforme o tamanho da entrada. Para arquivos grandes como Imagem (154 KB) e PDF (193 KB), tanto o AES quanto o 3DES consumiram aproximadamente 309–389 KB de memória, refletindo o uso intensivo de buffer durante a cifragem. Para entradas pequenas como ”Texto curto”, o uso de memória foi muito reduzido: RSA com apenas 3.1 KB e 3DES com 2.8 KB. Aparentemente, o AES, com 444 KB, tem uma sobrecarga interna maior, mesmo para dados pequenos.

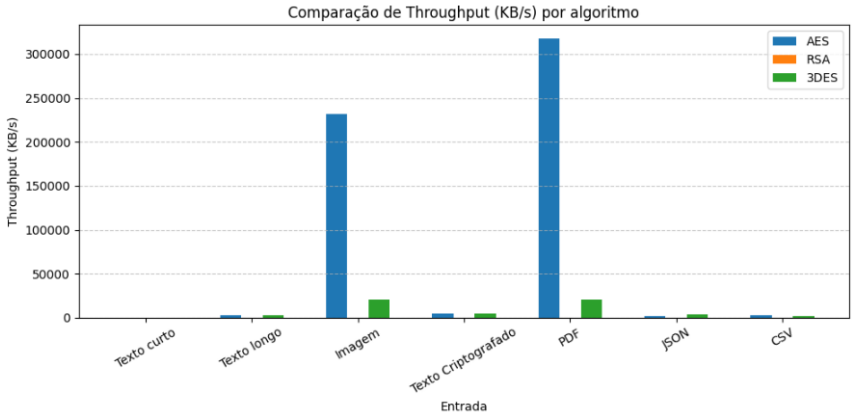

Figure 3. Comparação de Throughput por Algoritmo

Em termos de *throughput*, o AES se destaca com folga. Nas entradas ”Imagem” e ”PDF”, seu desempenho atingiu 231.845 KB/s e 317.301 KB/s, respectivamente. Já o 3DES, embora mais lento, ainda manteve valores expressivos, como 20.268 KB/s (Imagem) e 20.409 KB/s (PDF). O RSA, testado apenas no ”Texto curto”, teve um *throughput* de apenas 0,1426 KB/s, comprovando sua limitação prática. Mesmo em entradas menores, como CSV e JSON, o AES manteve *throughput* superior a 2.300 KB/s, o que o torna altamente eficiente em aplicações que exigem velocidade.

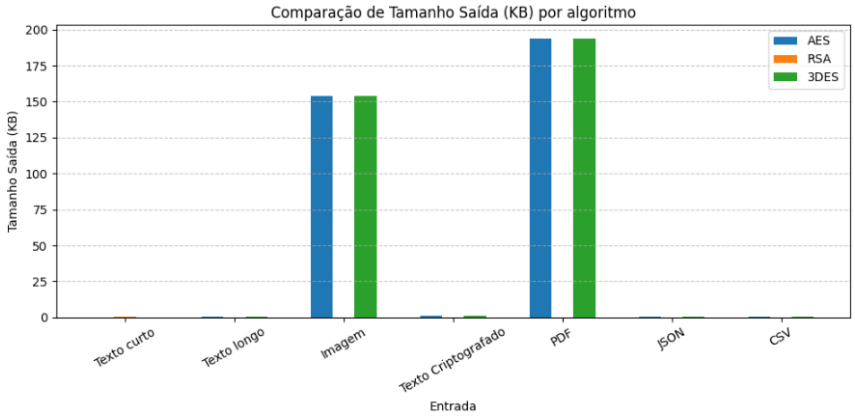

Figure 4. Comparação de Entropia por Algoritmo

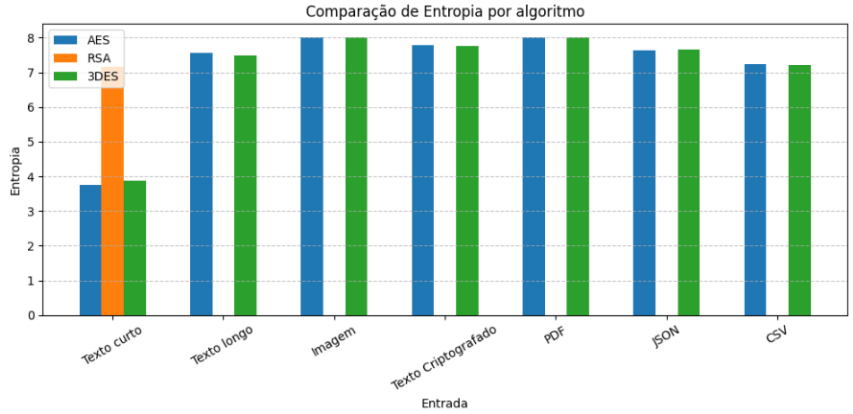

Figure 5. Comparação do Tamanho da Saída por Algoritmo

O tamanho da saída criptografada, como esperado, corresponde diretamente ao tamanho da entrada (com pequenas variações por conta de padding e modo de operação). Por exemplo, um arquivo PDF de 193,8 KB gerou uma saída do mesmo tamanho. Isso é importante porque mostra que, exceto pelo overhead da criptografia (como blocos ou cabeçalhos), o impacto no tamanho dos dados é geralmente mínimo. Vale destacar que o RSA também apresenta limitação no tamanho da saída, além da entrada, sendo mais adequado à criptografia de chaves ou pequenos trechos de dados.

6. Conclusão

A partir dos dados analisados, fica evidente que o AES é o algoritmo mais equilibrado, entregando alta performance, uso eficiente de memória e alto grau de entropia. O 3DES, embora seguro, apresenta desempenho inferior tanto em tempo quanto em *throughput*. O RSA, por sua natureza assimétrica, tem uso restrito a dados pequenos e não é viável para dados de maior volume. Essa análise corrobora a prática comum em sistemas criptográficos reais, nos quais o RSA é utilizado para cifrar apenas chaves de sessão, enquanto algoritmos simétricos como o AES são aplicados à criptografia de dados em larga escala.

Com base nos resultados obtidos, é possível estabelecer uma correlação clara entre o desempenho dos algoritmos e seus usos mais apropriados em aplicações práticas:

- O AES demonstrou ser o algoritmo mais eficiente entre os testados. Ele alia velocidade de execução, bom uso de memória, alto *throughput* e elevada entropia, tornando-se ideal para aplicações que exigem criptografia de grandes volumes de dados, como arquivos, bancos de dados, transmissões em tempo real e sistemas embarcados. Sua robustez e desempenho justificam seu uso extensivo em padrões como o TLS, IPsec e criptografia de disco.
- O 3DES, apesar de ainda oferecer uma segurança aceitável, teve desempenho inferior ao AES em todos os aspectos avaliados — principalmente em velocidade e *throughput*. Seu uso hoje é considerado obsoleto para novos sistemas, sendo mais encontrado em sistemas legados que ainda não migraram para padrões mais modernos. É adequado apenas em contextos com restrições tecnológicas ou necessidade de compatibilidade com sistemas antigos.
- O RSA, como esperado de um algoritmo de criptografia assimétrica, mostrou-se inadequado para criptografar grandes volumes de dados, tanto pelo tempo de execução quanto pelo baixo *throughput* e pelas limitações de tamanho de entrada. Isso reforça seu uso típico em aplicações como: troca segura de chaves simétricas, assinatura digital e infraestruturas de chave pública.

Portanto, cada algoritmo se mostra mais eficiente dentro de seu domínio e a escolha deve, portanto, considerar não apenas o nível de segurança, mas também os requisitos de desempenho, volume de dados e o contexto de uso. Esta análise empírica confirma o que já é amplamente adotado na indústria: o uso combinado de criptografia simétrica (como AES) para dados e assimétrica (como RSA) para chaves — estratégia que equilibra segurança e desempenho em sistemas modernos.

7. Referências

References

[Andrade and dos Santos Silva 2012] Andrade, R. S. and dos Santos Silva, F. (2012). Algoritmo de criptografia rsa: análise entre a segurança e velocidade. *Eventos Pedagógicos*, 3(3):438–457.

[de Oliveira and de Freitas Júnior 2004] de Oliveira, R. R. and de Freitas Júnior, J. L. (2004). Implementação, comparação e análise dos algoritmos idea e des. *ERMACS*.

[Lunkes and de Oliveira 2025] Lunkes, A. D. L. Z. and de Oliveira, F. B. (2025). Análise da complexidade dos algoritmos de criptografia homomórfica. *Ciência e Natura*, 47:e87825–e87825.

[Martínez-Plumed et al. 2021] Martínez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernández-Orallo, J., Kull, M., Lachiche, N., Ramírez-Quintana, M. J., and Flach, P. (2021). Crisp-dm twenty years later: From data mining processes to data science trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 33(8):3048–3061.

[Oliveira 2012] Oliveira, R. R. (2012). Criptografia simétrica e assimétrica-os principais algoritmos de cifragem. *Segurança Digital [Revista online]*, 31:11–15.

[Stallings 2014] Stallings, W. (2014). *Criptografia E Segurança De Redes: PRINCÍPIOS E PRÁTICAS*. PEARSON BRASIL.

[Voitechen 2015] Voitechen, D. A. (2015). Análise e comparação de algoritmos para criptografia de imagens. B.S. thesis, Universidade Tecnológica Federal do Paraná.

A. Tabela de Dados

Table 1. Resultados das métricas por algoritmo e tipo de entrada

Entrada	Alg	Tempo (s)	Memória (KB)	Throughput (KB/s)	Entropia	Tamanho (KB)
Texto curto	AES	0.0362	444.98	0.216	3.7500	0.0078
Texto curto	RSA	0.0548	3.13	0.1426	7.1559	0.0078
Texto curto	3DES	0.0004	2.86	18.3240	3.8750	0.0078
Texto longo	AES	0.0002	2.15	2789.77	7.5522	0.4346
Texto longo	RSA	-	-	-	-	0.4346
Texto longo	3DES	0.0002	2.40	2733.75	7.4811	0.4346
IMG	AES	0.0007	309.52	231845.75	7.9989	154.1348
IMG	RSA	-	-	-	-	154.1348
IMG	3DES	0.0076	309.76	20268.49	7.9989	154.1348
Texto cript.	AES	0.0002	2.98	4417.61	7.7829	0.8750
Texto cript.	RSA	-	-	-	-	0.8750
Texto cript.	3DES	0.0002	3.19	5202.57	7.7682	0.8750
PDF	AES	0.0006	388.86	317301.37	7.9988	193.8477
PDF	RSA	-	-	-	-	193.8477
PDF	3DES	0.0095	389.15	20409.25	7.9990	193.8477
JSON	AES	0.0002	2.32	2305.09	7.6465	0.5713
JSON	RSA	-	-	-	-	0.5713
JSON	3DES	0.0001	2.57	3812.66	7.6503	0.5713
CSV	AES	0.0001	1.66	2965.33	7.2295	0.2627
CSV	RSA	-	-	-	-	0.2627
CSV	3DES	0.0001	1.90	2160.31	7.2252	0.2627

B. Ambiente de Testes

O ambiente de testes utilizado para análise foi um *Notebook* no *Google Colab*, a seguir está o link: Ambiente de Testes